Please type a sign(+) inside this box → ⊞

PTO/SB/05 (2/98)
Approved for use through 09/30/2000 OMB 0651-0032
Patent and Trademark Office. U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR § 1.53(b))

Attorney Docket No.	*1778.0100002*
First Inventor or Application Identifier	*Timothy J. Van Hook*
Title	*Alignment and Ordering of Vector Elements for Single Instruction Multiple Data Processing*
Express Mail Label No.	

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. ☒ * Fee Transmittal Form (*e.g.*, PTO/SB/17)
(Submit an original, and a duplicate for fee processing)
2. ☒ Specification [Total Pages 42]
(preferred arrangement set forth below)
- Descriptive title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings *(if filed)*
- Detailed Description
- Claim(s)
- Abstract of the Disclosure
3. ☒ Drawing(s) (35 U.S.C. 113) [Total Sheets 17]
4. ☒ Oath or Declaration [Total Pages 15]
   a. ☐ Newly executed (original or copy)
   b. ☒ Copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional with Box 17 completed)* ***[Note Box 5 below]***
      I. ☐ DELETION OF INVENTOR(S)
      Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR §§ 1.63(d)(2) and 1.33(b).
5. ☒ Incorporation By Reference *(useable if Box 4b is checked)*
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein
6. ☐ Microfiche Computer Program *(Appendix)*
7. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   a. ☐ Computer Readable Copy
   b. ☐ Paper Copy (identical to computer copy)
   c. ☐ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. ☐ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3 73(b) Statement (when there is an assignee) ☐ Power of Attorney
10. ☐ English Translation Document *(if applicable)*
11. ☐ Information Disclosure Statement (IDS)/PTO-1449 ☐ Copies of IDS Citations
12. ☒ Preliminary Amendment
13. ☒ Two (2) Return Receipt Postcards (MPEP 503) *(Should be specifically itemized)*
14. ☐ *Small Entity Statement(s) *(PTO/SB/09-12)* ☐ Statement filed in prior application, Status still proper and desired
15. ☐ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
16. ☒ Other: 37 C F R § 1 136(a)(3) Authorization
☐ Other

**NOTE FOR ITEMS 1 & 14 IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F R § 1 27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. §1 28)*

17. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment*

☒ Continuation ☐ Divisional ☐ Continuation-in-Part (CIP) of prior application No: 09/263,798

Prior application information: *Examiner* *S. Whitmore* *Group/Art Unit.* 2783

18. CORRESPONDENCE ADDRESS

☐ ***Customer Number or Bar Code Label*** ***(Insert Customer No. or Attach bar code label here)*** *or* ☒ ***Correspondence address below***

NAME	STERNE, KESSLER, GOLDSTEIN & FOX P L L.C				
	Attorneys at Law				
ADDRESS	Suite 600, 1100 New York Avenue, N.W.				
CITY	Washington	*STATE*	DC	*ZIP CODE*	20005-3934
COUNTRY	USA	*TELEPHONE*	(202) 371-2600	*FAX*	(202) 371-2540

NAME (Print/Type)	Michael B. Ray	*Registration No. (Attorney/Agent)*	33,997
SIGNATURE		*Date*	9/15/00

Burden Hour Statement: this form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

A

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.
ATTORNEYS AT LAW
1100 NEW YORK AVENUE, N.W., SUITE 600
WASHINGTON, D C 20005-3934

www skgf.com
PHONE (202) 371-2600 FACSIMILE (202) 371 2540

September 15, 2000

WRITER'S DIRECT NUMBER:
(202) 371-2569
INTERNET ADDRESS:
MRAY@SKGF COM

Commissioner for Patents
Washington, D.C. 20231

Box Patent Application

Re: U.S. Non-Provisional Utility Patent Application under 37 C.F.R. § 1.53(b)
Continuation of U.S. Patent Appl. No. 09/263,798; filed March 5, 1999
Appl. No. To be assigned; Filed: Herewith
For: **Alignment and Ordering of Vector Elements for Single Instruction Multiple Data Processing**
Inventors: Van Hook *et al.*
Our Ref: 1778.0100002 (0055.20US)

Sir:

The following documents are forwarded herewith for appropriate action by the U.S. Patent and Trademark Office:

1. PTO Utility Patent Application Transmittal (Form PTO/SB/05);

2. U.S. Utility Patent Application entitled:

Alignment and Ordering of Vector Elements for Single Instruction Multiple Data Processing

and naming as inventors:

Timothy J. Van HOOK
Peter HSU
William HUFFMAN
Henry P. MORETON
Earl KILLIAN

the application consisting of:

a A specification containing:

i. 30 pages of description prior to the claims;

ii. 11 pages of claims (48 claims);

iii. a one (1) page abstract;

b. 17 sheets of drawings: (Figures 1-5, 6A-6B, 7, 8A-8H, 9, 10A-10H);

c. A copy of the executed combined Declaration and Power of Attorney, as originally filed in U.S. Appl. No. 08/947,649;

3. Authorization to Treat a Reply As Incorporating An Extension of Time Under 37 C.F.R. § 1.136(a)(3) (*in duplicate*);

4. A Preliminary Amendment;

5. Letter to PTO Draftsman: Submission of Formal Drawings;

6. 17 sheets of formal drawings (Figures 1-5, 6A-6B, 7, 8A-8H, 9,10A-10H), approval of which is respectfully requested; and

7. Two (2) return postcards.

It is respectfully requested that, of the two attached postcards, one be stamped with the filing date of these documents and returned to our courier, and the other, prepaid postcard, be stamped with the filing date and unofficial application number and returned as soon as possible.

This application is being filed under 37 C.F.R. § 1.53(b) without filing fee.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.

Michael B. Ray
Attorney for Applicants
Registration No. 33,997

MBR/MPT/bnp
Enclosures

P:\USERS\BPORTER\MBR\1778 0100002 PTO Cover Letter (0055 20US)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	
Van Hook *et al*	Art Unit: To be assigned
Appl. No.: To be assigned	Examiner: To be assigned
Filed: Herewith	Atty. Docket: 1778.0100002 (0055.20US)
For: **Alignment and Ordering of Vector Elements for Single Instruction Multiple Data Processing**	

Preliminary Amendment

Commissioner for Patents
Washington, D.C. 20231

Sir:

Prior to examination of the above-captioned application, Applicant submits the following Amendments and Remarks.

In the Specification:

Page 1, between lines 3 and 4, please insert the following:

--This application is a continuation of U.S. Patent Application No. 09/263,798, filed March 5, 1999, which is a continuation of U.S. Patent Application No. 08/947,649, filed October 9, 1997, now U.S. Patent No. 5,933,650, issued August 3, 1999.--

In the Claims:

Please cancel claims 2-48 without prejudice or disclaimer.

Remarks

Upon entry of the foregoing amendments, claim 1 is pending in the application. Claims 2-48 are sought to be canceled without prejudice or disclaimer. The above amendments are to matters of form only and their entry is respectfully requested. These changes are believed to introduce no new matter, and their entry is respectfully requested.

The Examiner is invited to telephone the undersigned representative if he believes that an interview might be useful for any reason.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.

Michael B. Ray
Attorney for Applicant
Registration No. 33,997

Date: 9/15/00

1100 New York Avenue, N.W.
Suite 600
Washington, D.C. 20005-3934
(202) 371-2600

MBR/MPT bnp
P \USERS\BPORTER\MBR\1778 0100002 Preliminary Amend (0055 20US)

UNITED STATES PATENT APPLICATION

FOR

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

Inventors:

Timothy J. van Hook

Peter Hsu

William Huffman

Henry P. Moreton

Earl Killian

Prepared by:

WAGNER, MURABITO & HAO

Two North Market Street

Third Floor

San Jose, California 95113

SGI-15-4-457.00 October 7, 1997

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

FIELD OF THE INVENTION

The present invention relates to the field of single instruction multiple data vector (SIMD) processing. More particularly, the present claimed invention relates to alignment and ordering vector elements for SIMD processing.

BACKGROUND ART

Today, most processors in microcomputer systems provide a 64-bit wide datapath architecture. The 64-bit datapath allows operations such as read, write, add, subtract, and multiply on the entire 64 bits of data at once. However, for many applications the types of data involved simply do not require the full 64 bits. In media signal processing (MDMX) applications, for example, the light and sound values are usually represented in 8, 12, 16, or 24 bit numbers. This is because people typically are not able to distinguish the levels of light and sound beyond the levels represented by these numbers of bits. Hence, data types in MDMX applications typically require less than the full 64 bits provided in the datapath in most computer systems.

To efficiently utilize the entire datapath, the current generation of processors typically utilizes a single instruction multiple data (SIMD) method. According to this method, a multitude of smaller numbers are packed into the 64 bit doubleword as elements, each of which is then operated on independently and in parallel. Prior Art Figure 1 illustrates an exemplary single instruction multiple data (SIMD) method. Registers, vs and vt, in a

processor are of 64-bit width. Each register is packed with four 16-bit data elements fetched from memory: register vs contains vs[0], vs[1], vs[2], and vs[3] and register vt contains vt[0], vt[1], vt[2], and vt[3]. The registers in essence contain a vector of N elements. To add elements of matching index, an add instruction adds, independently, each of the element pairs of matching index from vs and vt. A third register, vd, of 64-bit width may be used to store the result. For example, vs[0] is added to vt[0] and its result is stored into vd[0]. Similarly, vd[1], vd[2], and vd[3] store the sum of vs and vd elements of corresponding indexes. Hence, a single add operation on the 64-bit vector results in 4 simultaneous additions on each of the 16-bit elements. On the other hand, if 8-bit elements were packed into the registers, one add operation performs 8 independent additions in parallel. Consequently, when a SIMD arithmetic instruction such as addition, subtraction, or multiply, is performed on the data in the 64-bit datapath, the operation actually performs multiple numbers of operations independently and in parallel on each of the smaller elements comprising the 64 bit datapath. In SIMD vector operation, processors typically require alignment to the data type size of 64-bit doubleword on a load. This alignment ensures that the SIMD vector operations occur on aligned boundaries of a 64-bit doubleword boundary.

Unfortunately, the elements within application data vectors are frequently not 64-bit doubleword aligned for SIMD operations. For example, data elements stored in a memory unit are loaded into registers in a chunk such as a 64-bit doubleword format. To operate on the individual elements, the elements are loaded into a register. The order of the elements in the register remain the same as the order in the original memory. Accordingly, the elements may not be properly aligned for a SIMD operation.

Traditionally, when elements are not aligned with a proper boundary as required for a SIMD vector operation, the non-aligned vector processing have typically been reduced to scalar processing. That is, operations took place one element at a time instead of simultaneous multiple operations. Consequently, SIMD vector operations lost parallelism and performance advantages when the vector elements were not properly aligned.

Furthermore, many media applications require a specific ordering for the elements within a SIMD vector. Since elements necessary for SIMD processing are commonly stored in multiple 64-bit doublewords with other elements, these elements need to be selected and assembled into a vector of desired order. For example, multiple channel data are commonly stored in separate arrays or interleaved in a single array. Processing the data requires interleaving or deinterleaving the multiple channels. Other applications require SIMD vector operations on transposed 2 dimensional arrays of data. Yet other applications reverse the order of elements in an array as in FFTs, DCTs, and convolution algorithms.

Thus, what is needed is a method for aligning and ordering elements for more efficient SIMD vector operations by providing computational parallelism.

SUMMARY OF THE INVENTION

The present invention provides alignment and ordering of vector elements for SIMD processing. The present invention is implemented in a computer system including a processor having a plurality of registers. In the alignment of vector elements for SIMD processing, one vector is loaded from a memory unit into a first register and another vector is loaded from the memory unit into a second register. The first vector contains a first byte of an aligned vector to be generated. Then, a starting byte specifying the first byte of an aligned vector is determined. Next, a vector is extracted from the first register and the second register beginning from the first bit in the first byte of the first register continuing through the bits in the second register. Finally, the extracted vector is replicated into a third register such that the third register contains a plurality of elements aligned for SIMD processing. In the ordering of vector elements for SIMD processing, a first vector is loaded from a memory unit into a first register and a second vector is loaded from the memory unit into a second register. Then, a subset of elements is selected from the first register and the second register. The elements from the subset are then replicated into the elements in the third register in a particular order suitable for subsequent SIMD vector processing.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

Prior Art Figure 1 illustrates an exemplary single instruction multiple data (SIMD) instruction method.

Figure 2 illustrates a block diagram of an exemplary computer system for implementing the present invention.

Figure 3 illustrates a block diagram of an exemplary datapath for aligning and ordering vector elements.

Figure 4 illustrates a block diagram of an alignment unit in a processor for aligning a vector of elements.

Figure 5 illustrates a flow diagram of the steps involved in extracting an aligned vector from two exemplary vectors.

Figure 6A illustrates a block diagram of a full byte-mode crossbar circuit used in generating a vector of elements from elements of two vector registers.

Figure 6B shows a more detailed diagram of the operation of an exemplary AND gate associated with element 7 in the first register, vs.

Figure 7 illustrates shuffle operations for ordering 8-bit elements in a 64-bit doubleword.

Figure 8A illustrates a block diagram of a shuffle operation, which converts four unsigned upper bytes (i.e., 8 bits) in a source register to four 16-bit halves in a destination register.

Figure 8B illustrates a block diagram of a shuffle operation, which converts a vector of unsigned low 4 bytes from a source register to four 16-bit halves in a destination register.

Figure 8C illustrates a block diagram of a shuffle operation, which converts a vector of signed upper 4 bytes from a source register to four 16-bit halves in a destination register by replicating the signs across the upper bytes in the halves.

Figure 8D illustrates a block diagram of a shuffle operation, which converts a vector of signed low 4 bytes from a source register to four 16-bit halves in a destination register by replicating the signs across the upper bytes in the halves.

Figure 8E illustrates a block diagram of a shuffle operation, which replicates the odd elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register.

Figure 8F illustrates a block diagram of a shuffle operation, which replicates the even elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register.

Figure 8G illustrates a block diagram of a shuffle operation, which replicates the upper 4 elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register.

Figure 8H illustrates a block diagram of a shuffle operation, which replicates the lower 4 elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register.

Figure 9 illustrates shuffle operations for ordering 16-bit elements in a 64-bit doubleword.

Figure 10A illustrates a block diagram of a shuffle operation, which replicates the upper 2 elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register.

Figure 10B illustrates a block diagram of a shuffle operation, which replicates the lower 2 elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register.

Figure 10C illustrates a block diagram of a shuffle operation, which replicates 2 odd elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register.

Figure 10D illustrates a block diagram of a shuffle operation, which replicates 2 even elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register.

Figure 10E illustrates a block diagram of a shuffle operation, which replicates even elements 0 and 2 from one source register into odd elements 1 and 3 in a destination vector register and further replicates odd elements 1 and 3 from another source register into the even elements 0 and 2, respectively, of the destination vector register.

Figure 10F illustrates a block diagram of a shuffle operation, which replicates even elements 0 and 2 from one source register into odd elements 3 and 1, respectively, in a destination vector register and further replicates odd elements 1 and 3 from another source register into the even elements 2 and 0, respectively, of the destination vector register.

Figure 10G illustrates a block diagram of a shuffle operation, which replicates the upper 2 elements of 4 16-bit elements from each of two source registers into a destination vector register.

Figure 10H illustrates a block diagram of a shuffle operation, which replicates the lower 2 elements of 4 16-bit elements from each of two source registers into a destination vector register.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the present invention, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details. In other instances well known methods, procedures, components, and circuits have not been described in detail so as not to unnecessarily obscure aspects of the present invention.

The present invention, a method for providing alignment and ordering of vector elements for single-instruction multiple-data (SIMD) processing, is described. The preferred embodiment of the present invention provides elements aligned and ordered for an efficient SIMD vector operation in a processor having 64-bit wide datapath within an exemplary computer system described below. Although such a datapath is exemplified herein, the present invention can be readily adapted to suit other datapaths of varying widths.

COMPUTER SYSTEM ENVIRONMENT

Figure 2 illustrates an exemplary computer system 200 comprised of a system bus 206 for communicating information, a processor 202 coupled with the bus 206 for processing information and instructions, a computer readable volatile memory unit 210 (e.g., random access memory, static RAM, dynamic RAM, etc.) coupled with the bus 206 for storing information and instructions for the processor 202, a computer readable non-volatile memory unit 208 (e.g., read only memory, programmable ROM, flash memory, EPROM, EEPROM, etc.) coupled with the bus 206 for storing static information and instructions

for the processor 202. A vector register file 204 containing a plurality of registers is included in the processor 202. In the present invention, the term vector register file 204 encompasses any register file containing a plurality of registers and as such is not limited to vector register files.

The computer system 200 of Figure 2 further includes a mass storage computer readable data storage device 212 (hard drive, floppy, CD-ROM, optical drive, etc.) such as a magnetic or optical disk and disk drive coupled with the bus 206 for storing information and instructions. Optionally, the computer system 200 may include a display device 214 coupled to the bus 206 for displaying information to the user, an alphanumeric input device 216 including alphanumeric and function keys coupled to the bus 206 for communicating information and command selections to the processor 202, a cursor control device 218 coupled to the bus 206 for communicating user input information and command selections to the processor 202, and a signal generating device 220 coupled to the bus 206 for communicating command selections to the processor 202.

According to an exemplary embodiment of the present invention, the processor 202 includes a SIMD vector unit that functions as a coprocessor for or as an extension of the processor 202. The SIMD vector unit performs various arithmetic and logical operations on each data element within a SIMD vector in parallel. The SIMD vector unit utilizes the register files of the processor 202 to hold SIMD vectors. The present invention may include one or more SIMD vector units to perform specialized operations such as arithmetic operations, logical operations, etc.

Figure 3 illustrates a block diagram of an exemplary datapath 300 for aligning and ordering vector elements. The datapath 300 includes a SIMD vector unit 302, an alignment unit 322, a register file 304, a crossbar circuit 314, and a vector load/store unit 302. The vector load/store unit 302 performs load and store functions. It loads a vector from memory into one of the registers in the register file 304. It also stores a vector from one of the registers in the register file 304 into main memory. The alignment unit 312 receives two vectors from two source registers such as vs 306 and vt 308. Then, the alignment unit 312 extracts an aligned vector from the two vectors and stores it into a destination register such as vd 310. The crossbar circuit 314 also receives two vectors two exemplary source registers, vs 306 and vt 308. The crossbar circuit 314 then selects a set of elements from the source registers and routes each of the elements in the selected set to a specified element in the exemplary destination register, vd 310. In an alternative embodiment, the crossbar circuit 314 may receive one vector from a single source register and select a set of elements from the vector. The data path 318 allows a result to be forwarded to the register file 304 or to the vector load/store unit to be stored into main memory.

The SIMD vector unit 302 represents a generic SIMD vector processing unit, which may be an arithmetic unit, logical unit, integer unit, etc. The SIMD vector unit 302 may receive either one or two vectors from one or two source registers. It should be appreciated that the present invention may include more than one SIMD vector unit performing various functions. The SIMD vector unit 302 may execute an operation specified in the instruction on each element within a vector in parallel.

The exemplary vector register file 304 is preferably comprised of 32 64-bit general purpose registers. To this end, the preferred embodiment of the present invention utilizes the floating point registers (FGR) of a floating point unit (FPU) in the processor as its vector registers. In this shared arrangement, data is moved between the vector register file 304 and a memory unit through the vector load/store unit 302. These load and store operations are unformatted. That is, no format conversions are performed and therefore no floating-point exceptions can occur due to these operations. Similarly, data is moved between the vector register file 304 and the alignment unit 312, the crossbar circuit 314, or the SIMD vector unit 316 without format conversions, and thus no floating-point exception occurs.

The present invention allows data types of 8-, 16-bit, 32-, or 64-bit fields. Hence, a 64-bit doubleword vector may contain 8 8-bit elements, 4 16-bit elements, 2 32-bit elements, or 1 64-bit element. According to this convention, vector registers of the present invention are interpreted in the following data formats: Quad Half (QH), Oct Byte (OB), Bi word (BW), and Long (L). In QH format, a vector register is interpreted as having 16-bit elements. For example, a 64-bit vector register is interpreted as a vector of 4 signed 16-bit integers. OB format interprets a vector register as being comprised of 8-bit elements. Hence, an exemplary 64-bit vector register is seen as a vector of 8 unsigned 8-bit integers. In BW format, a vector register is interpreted as having 2 32-bit elements. L format interprets a vector register as having a 64-bit element. These data types are provided to be adaptable to various register sizes of a processor. As described above, data format conversion is not necessary between these formats and floating-point format.

According to a preferred embodiment of the present invention, exemplary source registers, vs and vt, are each used to hold a set of vector elements. A third exemplary vector register, vd, is created from the source registers and holds a set of elements selected from the source registers. Although the registers, vs, vt, and vd, are used to associate vector registers with a set of vector elements, other vector registers are equally suitable for present invention.

LOAD/STORE INSTRUCTIONS

The load and store instructions of the present invention use a special load/store unit to load and store a 64-bit doubleword between a register in a register file such as an FPR and a memory unit. The doubleword is loaded through an exemplary load/store unit 302 illustrated above in Figure 3. The load/store unit performs loading or storing of a doubleword with upper 61 bits of an effective address. The lowest 3 bits specify a byte address within the 64-bit doubleword for alignment.

According to a preferred embodiment, an effective address is formed by adding the contents of an index value in a general purpose register (GPR) to a base address in another GPR. The effective address is doubleword aligned. During the loading process, the last three bits of the effective address are ignored by treating these bits as 0s. Hence, the effective address is comprised of bits 3 to 63. The three bits from 0 to 2 contain the byte address for accessing individual bytes within a doubleword and are ignored by treating the three bits as 0s. If the size of a register in a register file is 64-bits, then the 64-bit data stored in memory at the effective address is fetched and loaded into the register. If on the other hand, the size of the register in the register file is 32-

bits, then the lower 32 bits of the data are loaded into the vector register and the upper 32 bits of the data are loaded into the next register in sequence. Hence, a pair of 32-bit registers are used to hold a 64-bit data from the memory.

Conversely, the store instruction stores a doubleword from a vector register such as an FPR to the memory while ignoring alignment. The store operation is carried out through the exemplary load/store unit 302 illustrated above in Figure 3. The contents of a 64-bit doubleword in FPR, fs, is stored at the memory location specified by the effective address. The contents of GPR index and GPR base are added to form the effective address. The effective address is doubleword aligned. The last three bits of the effective address are ignored.

The effective address is formed by adding the contents of an index value in a general purpose register (GPR) to a base address in another GPR while ignoring the lowest three bits of the effective address by interpreting them as 0s. That is, the effective address is comprised of bits 3 to 63. The ignored three bits contain the byte address for accessing individual bytes within a doubleword. If the size of a vector register is 64-bits, then the content of the vector register is stored into memory. If on the other hand, the size of a vector register is 32-bits, then the lower 32 bits of the data are concatenated with the upper 32 bits of the data contained in the next register in sequence. Then, the concatenated 64-bit doubleword is stored into memory at the address specified by the effective address.

ALIGNMENT INSTRUCTION

The present alignment instruction operates on two 64-bit doublewords loaded into two registers from memory by issuing two load instructions. One doubleword is loaded into a first register (vs) and the other doubleword is loaded into a second register (vt). The alignment instruction generates a 64-bit doubleword vector in a third register (vd) aligned for a SIMD vector operation. Preferably, an alignment unit performs alignment of a vector by funnel shift to extract an aligned 64-bit vector of elements from the two 64-bit registers.

Figure 4 illustrates a block diagram of an alignment unit in a processor for aligning a vector of elements. The vector load/store unit 404 loads two vectors from main memory 402 into two vector registers, vs and vt, in a register file 408. The alignment unit 410 receives the two vectors in the vector registers, vs and vt, and extracts a byte aligned vector. Three control lines 412 representing three bits for the byte address controls the byte alignment performed through the alignment unit 410. The aligned vector is then forwarded to an exemplary vector register, vd, in the register file.

The alignment of a vector is dependent on a byte ordering mode of a processor. Byte ordering within a larger data size such as a 64-bit doubleword may be configured in either big-endian or little-endian order. Endian order refers to the location of byte 0 within a multi-byte data. A processor according to the present invention may be configured as either a big-endian or little-endian system. For example, in a little-endian system, byte 0 is the least significant (i.e., rightmost) byte. On the other hand, in a big-endian system, byte 0 is the most significant (i.e., leftmost) byte. In the present invention, an

exemplary processor uses byte addressing for a doubleword access, which is aligned on a byte boundary divisible by eight (i.e., 0, 8, 16, ..., 56). Hence, a 64-bit doubleword loaded into a register in a processor is byte-aligned in either a big-endian or a little-endian mode. For a little-endian mode processor, the starting (i.e., first) byte for a vector to be extracted lies in the second vector register. Conversely for a big-endian mode processor, the starting (i.e., first) byte for the vector resides in the first vector register.

Figure 5 illustrates a flow diagram of the steps involved in extracting an aligned vector from two exemplary vectors. In step 502, two 64-bit doublewords are loaded from a memory unit into two 64-bit registers. One 64-bit doubleword is loaded into a first register and the other 64-bit doubleword in memory is loaded into the second register. Preferably, the former doubleword and the next doubleword are stored in contiguous memory space and their starting addresses differ by 64-bits or 8 bytes. The loading of the doublewords are accomplished through a load/store unit according to the load instruction described above.

The starting byte address of the aligned vector to be extracted is then determined in step 704. According to the preferred embodiment, the register and vector are all 64-bit wide. Since a 64-bit doubleword contains 8 bytes, three bits are needed to specify all the byte positions in a 64-bit doubleword. Hence, the preferred embodiment uses 3 bits to specify the position of the starting byte address in a 64-bit vector.

In one embodiment of the present invention, an alignment instruction provides an immediate, which is a constant byte address within a

doubleword. Preferably, the immediate consists of 3 bits for specifying a constant byte address to a byte among 8 bytes each in the first register (i.e., little-endian mode processor) and the second register (i.e., bit-endian mode processor). This alignment instruction performs a constant alignment of a vector. The align amount is computed by masking the immediate, then using that value to control a funnel shift of vector vs concatenated with vector vt. The operands can be in the QH, OB, or BW format.

In an alternative embodiment, the alignment instruction provides a variable byte addressing by specifying an address of a general purpose register (GPR) containing the starting byte address in the first register. This instruction accesses the GPR by using the address provided in the alignment instruction. Then, the instruction extracts the lower 3 bits in the GPR to obtain the starting byte address in the first register (i.e., little-endian mode) or the second register (i.e., big-endian mode). The align amount is computed by masking the contents of GPR, rs, then using that value to control a funnel shift of vector vs concatenated with vector vt. The operands can be in QH, OB, or BW format.

After determining the starting byte address in step 504 of the flowchart in Figure 5, the first bit of the starting byte address is determined in step 506 by multiplying the starting byte address by 8. For example, if the starting byte address were 3, the first bit of the starting byte address is 3*8 or 24. Then in step 508, a 64-bit doubleword is extracted by concatenating from the first bit at the starting byte address in one register continuing through the other register. This concatenation is accomplished by funnel shifting from the first bit of the starting byte. Specifically, the first register is assigned bit positions from 0 to

63. The second register is assigned the next 64 bit positions from 64 to 127. The extraction scheme depends on the byte ordering modes. A variable s, representing the first bit position at the starting byte address, can be used to simplify the illustration of the differences between the byte ordering modes. In a big-endian byte mode, the concatenation occurs from bit position 127-s to 64-s. Conversely, in a little-endian bye mode, the concatenation occurs from bit position s through 63+s.

Then in step 510, the extracted vector is replicated into a destination register in the register file for SIMD vector processing. In an alternative, embodiment, the extracted vector may be stored into the memory unit for later use. The process then terminates in step 512.

SHUFFLE INSTRUCTION

The shuffle instruction according to the present invention provides a vector of ordered elements selected from either one or two other vector registers. One or more load/store instructions are used to load the vector(s) into registers for shuffle operation. One embodiment uses a full byte-mode crossbar to generate a vector of elements selected from the elements of two other exemplary vectors. That is, selected elements of the exemplary vectors, vs and vt, are merged into a new exemplary vector, vd. The new vector, vd, contains elements aligned for SIMD operation. Alternatively, a plurality of shuffle operations may be carried out to arrange the elements in a desired order for SIMD vector processing.

Figure 6A illustrates a block diagram of a full byte-mode crossbar circuit 600 used in generating a vector of elements from elements of two registers.

First, two vectors from a memory unit are loaded into two exemplary registers in a processor; the elements of the first vector are loaded into the first register, vs 602, and the elements of the second vector are loaded into the second register, vt 604. The elements of these two vector registers, vs 602 and vt 604, serve as source elements. The crossbar circuit 600 receives as input each of the elements from the two vector registers in parallel. A set of control lines 608 is coupled to the crossbar circuit 600 to relay a specific shuffle instruction operation. The shuffle instruction operation encodes a destination element for each of the selected source elements. In response to the specific shuffle instruction operation signals, the crossbar circuit 600 selects a set of elements from the two registers, vs 602 and vt 604, and routes or replicates each element to its associated destination element in an exemplary destination register, vd 606.

In addition, the present invention allows zeroing and sign extension of elements. For example with reference to Figure 6A, the present invention provides either zeroing or sign extension for each element in the first register, vs 602. In addition to providing the entire bits to the crossbar circuit 600, elements 0 through 7 in the first register, vs 602, provides their corresponding sign bits 612, 614, 616, 618, 620, 622, 624, and 626 (612 through 626) to the associated AND gates 628, 630, 632, 634, 636, 638, 640, and 642 (628 through 642). Each of the AND gates 628 through 642 also receives as the other input, a control signal 610, which originate from a specific shuffle instruction for specifying either zeroing or sign extension mode.

Figure 6B shows a more detailed diagram of the operation of the exemplary AND gate 628 associated with element 7 in the first register, vs 602.

The AND gate 628 receives a single sign bit 612 from the most significant bit in the element 7 of the first register, vs 602. The AND gate 628 also receives the control signal 610. To provide zeroing for element 7 for example, the control signal 610 inputs a 0 into the AND gate 628. In this case, the output 652 at the AND gate 628 is 0 no matter what the input is at the sign bit 612. On the other hand, when the control signal is 1, the AND gate 628 generates the sign bit 612 as the output 652, whatever the sign is. In both cases of zeroing and sign extension, the output 652 is routed to a plurality of output lines 654 for replicating the output signal into an appropriate width. Preferably, the output lines 654 matches the number of bits in each element in the first register, vs 602. The crossbar circuit 600 accepts the signals on these output lines 652 and uses these signals to zero or sign extend element 7 when necessary according to a shuffle instruction. The AND gates for the other elements 0 to 6 operate in a similar manner to provide zeroing and sign extension bit signals to the crossbar circuit 600.

The preferred embodiment of the present invention operates on vectors of elements in a preferred OB or QH mode. In an OB mode, a 64-bit doubleword vector is interpreted as having 8 8-bit elements. In a QH mode, the 64-bit vector is treated as containing 4 16-bit elements. For example, in OB mode, the crossbar circuit 600 selects, in parallel, as source elements eight 8-bit elements among the elements in the registers vs 602 and vt 604. Each of the eight elements is then replicated or routed into a particular destination element in the destination vector register, vd 606. In QH mode, the crossbar circuit selects four 16-bit elements and replicates or routes each element into a particular destination element in the destination register. Those skilled in the art will appreciate that the crossbar circuit represents one embodiment of

the present invention in implementing the shuffle instruction operations. A crossbar circuit is well known the art and is commonly used in conjunction with vector processing units.

Figure 7 illustrates shuffle operations for ordering 8-bit elements in a 64-bit doubleword. Each row represents the destination vector register, vd, comprised of 8 elements, vd[0] to vd[7]. The first row 702 is comprised of placeholders to indicate the 8 elements. Below the first row 702 are 8 different shuffle operations in OB mode as indicated by the content of destination vector register, vd, for each row 704 to 718. These shuffle operations in OB mode are illustrated in Figures 8A through 8H.

Figure 8A illustrates a block diagram of a shuffle operation, which converts four unsigned upper bytes (i.e., 8 bits) in a source register to four 16-bit halves in a destination register. This shuffle operation, represented by mnemonic UPUH.OB, selects the upper 4 8-bit elements in an exemplary vector register, vs. The selected elements vs[4], vs[5], vs[6], and vs[7] are replicated into destination elements vd[0], vd[2], vd[4], and vd[6], respectively. The odd elements of the destination vector register vd[1], vd[3], vd[5], and vd[7] are zeroed.

Figure 8B illustrates a block diagram of a shuffle operation, which converts a vector of unsigned low 4 bytes in a register to 16-bit halves. This shuffle operation, represented by mnemonic UPUL.OB, selects the lower 4 8-bit elements in an exemplary vector register, vs. The selected elements vs[0], vs[1], vs[2], and vs[3] are replicated into destination elements vd[0], vd[2],

vd[4], and vd[6], respectively. The odd elements of the destination vector register vd[1], vd[3], vd[5], and vd[7] are zeroed.

Figure 8C illustrates a block diagram of a shuffle operation, which converts a vector of signed upper 4 bytes in a register to 16-bit halves. This shuffle operation, represented by mnemonic UPSH.OB, selects the upper 4 8-bit elements in an exemplary vector register, vs. The selected elements vs[4], vs[5], vs[6], and vs[7] are replicated into destination elements vd[0], vd[2], vd[4], and vd[6], respectively. The odd elements of the destination vector register vd[1], vd[3], vd[5], and vd[7] replicates the sign bits of the selected elements vs[4], vs[5], vs[6], and vs[7], respectively.

Figure 8D illustrates a block diagram of a shuffle operation, which converts a vector of signed low 4 bytes in a register to 16-bit halves. This shuffle operation, represented by mnemonic UPSL.OB, selects the lower 4 8-bit elements in an exemplary vector register, vs. The selected elements vs[0], vs[1], vs[2], and vs[3] are replicated into destination elements vd[0], vd[2], vd[4], and vd[6], respectively. The odd elements of the destination vector register vd[1], vd[3], vd[5], and vd[7] replicates the sign bits of the selected elements vs[0], vs[1], vs[2], and vs[3], respectively.

Figure 8E illustrates a block diagram of a shuffle operation, which replicates the odd elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic PACH.OB, selects the odd elements of 8 8-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[1], vs[3], vs[5], and vs[7] are

replicated into destination elements vd[4], vd[5], vd[6], and vd[7], respectively. The elements vt[1], vt[3], vt[5], and vt[7] from the vector register vt are replicated into destination elements vd[0], vd[1], vd[2], and vd[3], respectively.

Figure 8F illustrates a block diagram of a shuffle operation, which replicates the even elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic PACL.OB, selects the even elements of 8 8-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[0], vs[2], vs[4], and vs[8] are replicated into destination elements vd[4], vd[5], vd[6], and vd[7], respectively. The elements vt[0], vt[2], vt[4], and vt[6] from the vector register vt are replicated into destination elements vd[0], vd[1], vd[2], and vd[3], respectively.

Figure 8G illustrates a block diagram of a shuffle operation, which replicates the upper 4 elements of 8 8-bit elements from each of two source registers into 8 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic MIXH.OB, selects the upper 4 elements of 8 8-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[4], vs[5], vs[6], and vs[7] are replicated into the odd elements of the destination vector register, namely vd[1], vd[3], vd[5], and vd[7], respectively. The elements vt[4], vt[5], vt[6], and vt[7] from the vector register vt are replicated into the even elements of the destination elements vd[0], vd[2], vd[4], and vd[6], respectively.

Figure 8H illustrates a block diagram of a shuffle operation, which replicates the lower 4 elements of 8 8-bit elements from each of two source

registers into 8 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic MIXL.OB, selects the lower 4 elements of 8 8-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[0], vs[1], vs[2], and vs[3] are replicated into the odd elements of the destination vector register, namely vd[1], vd[3], vd[5], and vd[7], respectively. The elements vt[0], vt[1], vt[2], and vt[3] from the vector register vt are replicated into the even elements of the destination elements vd[0], vd[2], vd[4], and vd[6], respectively.

A shuffle instruction operating in QH mode generates a new vector of elements for two types of operations. The first type of operation creates a vector of new data sizes by converting data sizes between 16-bit elements and 32-bit elements in a vector. The second type creates a new vector of elements drawn from two other vectors. The present exemplary data type conversion operations enable a larger range of computational data format than their storage format, such as 32 bit computation on 16 bit numbers. In addition, the present embodiment operations allow conversion of a data set from a smaller range format to a larger range format or vice versa as between 16 and 32 bit data.

Figure 9 illustrates shuffle operations for ordering 16-bit elements in a 64-bit doubleword. Each row represents the destination vector register, vd, comprised of 4 elements, vd[0] to vd[7]. The first row 902 is comprised of placeholders to indicate the 4 elements. Below the first row 902 are 4 different shuffle operations in QH mode as indicated by the content of destination vector register, vd, for each row 904 to 918. These shuffle operations in QH mode are illustrated in Figures 10A through 10H.

Figure 10A illustrates a block diagram of a shuffle operation, which replicates the upper 2 elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic MIXH.QH, selects the upper 2 elements of 4 16-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[2] and vs[3] are replicated into the odd elements of the destination vector register, namely vd[1] and vd[3], respectively. The elements vt[2] and vt[3] from the vector register vt are replicated into the even elements of the destination elements vd[0] and vd[2], respectively.

Figure 10B illustrates a block diagram of a shuffle operation, which replicates the lower 2 elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic MIXL.QH, selects the lower 2 elements of 4 16-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[0] and vs[1] are replicated into the odd elements of the destination vector register, namely vd[1] and vd[3], respectively. The elements vt[0] and vt[1] from the vector register vt are replicated into the even elements of the destination elements vd[0] and vd[2], respectively.

Figure 10C illustrates a block diagram of a shuffle operation, which replicates 2 odd elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic PACH.QH, selects the 2

odd elements of 4 16-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[1] and vs[3] are replicated into the upper 2 elements of the destination vector register, namely vd[2] and vd[3], respectively. The elements vt[1] and vt[3] from the vector register vt are replicated into the lower 2 elements of the destination elements vd[0] and vd[1], respectively.

Figure 10D illustrates a block diagram of a shuffle operation, which replicates 2 even elements of 4 16-bit elements from each of two source registers into 4 elements in a destination vector register. This shuffle operation, represented by an exemplary mnemonic PACL.QH, selects the 2 even elements of 4 16-bit elements in exemplary source vector registers, vs and vt. The elements selected from vs, namely vs[0] and vs[2] are replicated into the upper 2 elements of the destination vector register, namely vd[2] and vd[3], respectively. The elements vt[0] and vt[2] from the vector register vt are replicated into the lower 2 elements of the destination elements vd[0] and vd[1], respectively.

Figure 10E illustrates a block diagram of a shuffle operation, which replicates even elements from one source register and odd elements from another source register into a destination vector register. This shuffle operation, represented by an exemplary mnemonic BFLA.QH, selects the 2 even elements of 4 16-bit elements from an exemplary source vector register, vs. The shuffle operation also selects the 2 odd elements of 4 16-bit elements from another exemplary source vector register, vt. The even elements selected from vs, namely vs[0] and vs[2] are replicated into the 2 odd elements of the destination vector register, namely vd[1] and vd[3], respectively. The

odd elements vt[1] and vt[3] from the vector register vt are replicated into the 2 even elements of the destination elements vd[0] and vd[1], respectively.

Figure 10F illustrates a block diagram of a shuffle operation, which replicates even elements from one source register and odd elements from another source register into a destination vector register. This shuffle operation, represented by an exemplary mnemonic BFLB.QH, selects the 2 even elements of 4 16-bit elements from an exemplary source vector register, vs. The shuffle operation also selects the 2 odd elements of 4 16-bit elements from another exemplary source vector register, vt. The even elements selected from vs, namely vs[0] and vs[2] are replicated into the 2 odd elements of the destination vector register in reverse order, namely vd[3] and vd[1], respectively. The odd elements vt[1] and vt[3] from the vector register vt are replicated into the 2 even elements of the destination elements in reverse order, namely vd[0] and vd[1], respectively.

Figure 10G illustrates a block diagram of a shuffle operation, which replicates the upper 2 elements of 4 16-bit elements from each of two source registers into a destination vector register. This shuffle operation, represented by an exemplary mnemonic REPA.QH, selects the upper 2 elements of 4 16-bit elements in exemplary source vector registers, vs and vt. The upper elements selected from vs, namely vs[2] and vs[3] are replicated into the upper elements of the destination vector register, namely vd[2] and vd[3], respectively. The upper elements vt[2] and vt[3] from the vector register vt are replicated into the lower elements of the destination elements vd[0] and vd[2], respectively.

Figure 10H illustrates a block diagram of a shuffle operation, which replicates the lower 2 elements of 4 16-bit elements from each of two source registers into a destination vector register. This shuffle operation, represented by an exemplary mnemonic REPB.QH, selects the lower 2 elements of 4 16-bit elements in exemplary source vector registers, vs and vt. The lower elements selected from vs, namely vs[0] and vs[1] are replicated into the upper elements of the destination vector register, namely vd[2] and vd[3], respectively. The lower elements vt[0] and vt[1] from the vector register vt are replicated into the lower elements of the destination elements vd[0] and vd[2], respectively.

The shuffle instructions allow more efficient SIMD vector operations. First, the shuffle operation creates a vector of new data sizes by converting between 8-bit elements and 16-bit elements in a vector. These data type conversions enable a larger range of computational data format than their storage format, such as 16 bit computation on 8 bit numbers. For example, these operations allow conversion of a data set from a smaller range format to a larger range format or vice versa as between 8 and 16 bit audio or video data.

Second, the shuffle operations are also useful in interleaving and deinterleaving data. For example, some applications store multiple channel data in separate arrays, or interleaved in a single array. These applications typically require interleaving or deinterleaving the multiple channels. In these applications, separate R, G, B, A byte arrays may be converted into an interleaved RGBA array by the following series of shuffle instructions:

```
MIXL.OB RGL, R, G          ; RGRGRGRG
```

```
MIXL.OB BAL, B, A              ; BABABABA
MIXH.OB RGH, R, G              ; RGRGRGRG
MIXH.OB BAH, B, A              ; BABABABA
MIXL.QS RGBALL, RGL, BAL       ; RGBARGBA
MIXH.QS RGBALH, RGL, BAL       ; RGBARGBA
MIXL.QS RGBAHL, RGH, BAH       ; RGBARGBA
MIXH.QS RGBAHH, RGH, BAH;      RGBARGBA
```

Conversely, an interleaved RGBA array may be deinterleaved into separate R, G, B, and A arrays by the following series of shuffle instructions:

```
PACL.OB    GA0GA1, RGBA0, RGBA1
PACH.OB    RB0RB1, RGBA0, RGBA1
PACL.OB    GA2GA3, RGBA2, RGBA3
PACH.OB    RB2RB3, RGBA2, RGBA3
PACL.OB    A0A1A2A3, GA0GA1, GA2GA3
PACH.OB    G0G1G2G3, GA0GA1, GA2GA3
PACL.OB    B0B1B2B3, RB0RB1, RB2RB3
PACH.OB    R0R1R2R3, RB0RB1, RB2RB3
```

Third, some algorithms operate on 2 dimensional arrays of data such as images. Such an array typically orders the elements of the array in a major axis, where the elements are consecutive, and a minor axis, where the elements are separated by the size of the major axis. Often, a transpose operation is performed on the 2 dimensional array by converting the major axis to minor axis and vice versa. A common example is a discrete cosine transformation (DCT) requiring transposing 8x8 block of array. In this example, the 8x8 block of array consists of following elements:

	d0	d1	d2	d3	d4	d5	d6	d7
s0	A0	B0	C0	D0	E0	F0	G0	H0
s1	A1	B1	C1	D1	E1	F1	G1	H1
s2	A2	B2	C2	D2	E2	F2	G2	H2
s3	A3	B3	C3	D3	E3	F3	G3	H3
s4	A4	B4	C4	D4	E4	F4	G4	H4
s5	A5	B5	C5	D5	E5	F5	G5	H5
s6	A6	B6	C6	D6	E6	F6	G6	H6
s7	A7	B7	C7	D7	E7	F7	G7	H7

The present invention can transpose the 8x8 transpose in OB mode in 24 instructions, of which 12 are shown as follows:

```
MIXH.OB t0, s0, s1      A0 A1 B0 B1 C0 C1 D0 D1
MIXH.OB t1, s2, s3      A2 A3 B2 B3 C2 C3 D2 D3
MIXH.OB t2, s4, s5      A4 A5 B4 B5 C4 C5 D4 D5
MIXH.OB t3, s6, s7      A6 A7 B6 B7 C6 C7 D6 D7
MIXH.QH u0, t0, t1      A0 A1 A2 A3 B0 B1 B2 B3
MIXH.QH u1, t2, t3      A4 A5 A6 A7 B4 B5 B6 B7
MIXH.QH u2, t0, t1      C0 C1 C2 C3 D0 D1 D2 D3
MIXH.QH u3, t2, t3      C4 C5 C6 C7 D4 D5 D6 D7
REPA.QH d0, u0, u1      A0 A1 A2 A3 A4 A5 A6 A7
REPB.QH d1, u0, u1      B0 B1 B2 B3 B4 B5 B6 B7
REPA.QH d2, u2, u3      C0 C1 C2 C3 C4 C5 C6 C7
REPB.QH d3, u2, u3      D0 D1 D2 D3 D4 D5 D6 D7

MIXL.OB t0, s0, s1      E0 E1 F0 F1 G0 G1 H0 H1
MIXL.OB t1, s2, s3      E2 E3 F2 F3 G2 G3 H2 H3
MIXL.OB t2, s4, s5      E4 E5 F4 F5 G4 G5 H4 H5
MIXL.OB t3, s6, s7      E6 E7 F6 F7 G6 G7 H6 H7
MIXL.QH u0, t0, t1      E0 E1 E2 E3 F0 F1 F2 F3
MIXL.QH u1, t2, t3      E4 E5 E6 E7 F4 F5 F6 F7
MIXL.QH u2, t0, t1      G0 G1 G2 G3 H0 H1 H2 H3
MIXL.QH u3, t2, t3      G4 G5 G6 G7 H4 H5 H6 H7
REPA.QH d0, u0, u1      E0 E1 E2 E3 E4 E5 E6 E7
REPB.QH d1, u0, u1      F0 F1 F2 F3 F4 F5 F6 F7
REPA.QH d2, u2, u3      G0 G1 G2 G3 G4 G5 G6 G7
REPB.QH d3, u2, u3      H0 H1 H2 H3 H4 H5 H6 H7
```

In another example, an exemplary 4x4 array block consists of following elements:

	d0	d1	d2	d3
s0	A	B	C	D
s1	E	F	G	H
s2	I	J	K	L
s3	M	N	O	P

A transpose operation of the 4x4 array block in QH mode uses 8 shuffle instructions as follows:

```
MIXH.QH t0, so, s1      A E B F
MIXH.QH t1, s2, s3      I M J N
```

```
REPA.QH d0, t0, t1      A E I M
REPB.QH d1, t0, t1      B F J N

MIXL.QH t0, s0, s1      C G D H
MIXL.QH t1, s2, s3      K O L P
REPA.QH d2, t0, t1      C G K O
REPB.QH d3, t0, t1      D H L P
```

The shuffle instructions such as BFLA and BFLB allow reversing the order of elements in an array, in pairs or groups of 4. Larger groups can be reordered by memory or register address because they are a multiple of 64 bit elements. Inverting the order of a large array can be accomplished by inverting each vector of 4 elements with BFLB and loading from or storing each doubleword to the mirrored address in the array. Similarly, a butterfly on a large array can be assembled from double word addressing and BFLA or BFLB operations on the addressed doublewords.

The present invention thus provides a method for providing element alignment and ordering for SIMD processing. While the present invention has been described in particular embodiments, it should be appreciated that the present invention should not be construed as being limited by such embodiments, but rather construed according to the claims below.

<u>CLAIMS</u>

What is claimed is:

1. In a computer system including a processor having a plurality of registers, a method for generating an aligned vector of first width from two second width vectors for single instruction multiple data (SIMD) processing, comprising the steps of:

loading a first vector from a memory unit into a first register, wherein the first vector contains a first byte of an aligned vector to be generated;

loading a second vector from the memory unit into a second register;

determining a starting byte in the first register wherein the starting byte specifies the first byte of an aligned vector;

extracting a first width vector from the first register and the second register beginning from the first bit in the first byte of the first register continuing through the bits in the second register; and

replicating the extracted first width vector into a third register such that the third register contains a plurality of elements aligned for SIMD processing.

2. The method as recited in Claim 1 further comprising the step of storing the aligned vector in the third register to the memory unit.

3. The method as recited in Claim 1, wherein the first width and second width are each 64 bits.

4. The method as recited in Claim 3, wherein the third register is comprised of 8 8-bit elements.

5. The method as recited in Claim 3, wherein the third register is comprised of 4 16-bit elements.

6. The method as recited in Claim 1, wherein the starting byte is specified as a constant in an alignment instruction.

7. The method as recited in Claim 1, wherein the starting byte is specified as a variable in a register in an alignment instruction.

8. The method as recited in Claim 1, wherein the first vector and the second vector are in contiguous location in the memory unit.

9. The method as recited in Claim 1, wherein the processor operates in a big-endian byte ordering mode.

10. The method as recited in Claim 1, wherein the processor operates in a little-endian byte ordering mode.

11. In a computer system including a processor having a plurality of registers, a method for generating an ordered set of elements in an N-bit vector from two sets of elements in two N-bit vectors for single instruction multiple data (SIMD) vector processing, said method comprising the steps of:

loading a first vector from a memory unit into a first register;

loading a second vector from the memory unit into a second register;

selecting a subset of elements from the first register and the second register; and

replicating the elements from the subset into the elements in the third register in a particular order suitable for subsequent SIMD vector processing.

12. The method as recited in Claim 11 further comprising the step of storing the elements in the third register to the memory unit.

13. The method as recited in Claim 11, wherein the first vector and the second vector are each comprised of 4 16-bit elements indexed from 0 to 3.

14. The method as recited in Claim 11, wherein the first vector and the second vector are each comprised of 8 8-bit elements indexed from 0 to 7.

15. The method as recited in Claim 13, wherein the subset is comprised of two elements from the first register and two elements from the second register.

16. The method as recited in Claim 14, wherein the subset is comprised of four elements from the first register and four elements from the second register.

17. The method as recited in Claim 13, wherein the subset is comprised of the elements 2 and 3 from the first register and the elements 2 and 3 from the second register.

18. The method as recited in Claim 17, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 2 of the second register;

the element 1 replicated from the element 2 of the first register;
the element 2 replicated from the element 3 of the second register; and
the element 3 replicated from the element 3 of the first register.

19. The method as recited in Claim 13, wherein the subset is comprised of the elements 0 and 1 from the first register and the elements 0 and 1 from the second register.

20. The method as recited in Claim 19, wherein the particular order of the elements in the third register comprises:
the element 0 replicated from the element 0 of the second register;
the element 1 replicated from the element 0 of the first register;
the element 2 replicated from the element 1 of the second register; and
the element 3 replicated from the element 1 of the first register.

21. The method as recited in Claim 13, wherein the subset is comprised of the elements 1 and 3 from the first register and the elements 1 and 3 from the second register.

22. The method as recited in Claim 21, wherein the particular order of the elements in the third register comprises:
the element 0 replicated from the element 1 of the second register;
the element 1 replicated from the element 3 of the second register;
the element 2 replicated from the element 1 of the first register; and
the element 3 replicated from the element 3 of the first register.

23. The method as recited in Claim 13, wherein the subset is comprised of the elements 0 and 2 from the first register and the elements 0 and 2 from the second register.

24. The method as recited in Claim 23, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 0 of the second register;
the element 1 replicated from the element 2 of the second register;
the element 2 replicated from the element 0 of the first register; and
the element 3 replicated from the element 2 of the first register.

25. The method as recited in Claim 13, wherein the subset is comprised of the elements 0 and 2 from the first register and the elements 1 and 3 from the second register.

26. The method as recited in Claim 25, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 1 of the second register;
the element 1 replicated from the element 0 of the first register;
the element 2 replicated from the element 3 of the second register; and
the element 3 replicated from the element 2 of the first register.

27. The method as recited in Claim 13, wherein the subset is comprised of the elements 0 and 2 from the first register and the elements 1 and 3 from the second register.

28. The method as recited in Claim 27, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 3 of the second register;

the element 1 replicated from the element 2 of the first register;

the element 2 replicated from the element 1 of the second register; and

the element 3 replicated from the element 0 of the first register.

29. The method as recited in Claim 13, wherein the subset is comprised of the elements 2 and 3 from the first register and the elements 2 and 3 from the second register.

30. The method as recited in Claim 29, wherein particular order of the elements in the third register comprises:

the element 0 replicated from the element 2 of the second register;

the element 1 replicated from the element 3 of the second register;

the element 2 replicated from the element 2 of the first register; and

the element 3 replicated from the element 3 of the first register.

31. The method as recited in Claim 13, wherein the subset is comprised of the elements 0 and 2 from the first register and the elements 0 and 1 from the second register.

32. The method as recited in Claim 31, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 0 of the second register;

the element 1 replicated from the element 1 of the second register;

the element 2 replicated from the element 0 of the first register; and

the element 3 replicated from the element 2 of the first register.

33. The method as recited in Claim 14, wherein the subset is comprised of the elements 1, 3, 5, and 7 from the first register and the elements 1, 3, 5, and 7 from the second register.

34. The method as recited in Claim 33, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 1 of the second register;
the element 1 replicated from the element 3 of the second register;
the element 2 replicated from the element 5 of the second register;
the element 3 replicated from the element 7 of the second register;
the element 4 replicated from the element 1 of the first register;
the element 5 replicated from the element 3 of the first register;
the element 6 replicated from the element 5 of the first register; and
the element 7 replicated from the element 7 of the first register.

35. The method as recited in Claim 14, wherein the subset is comprised of the elements 0, 2, 4, and 6 from the first register and the elements 0, 2, 4, and 6 from the second register.

36. The method as recited in Claim 35, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 0 of the second register;
the element 1 replicated from the element 2 of the second register;
the element 2 replicated from the element 4 of the second register;
the element 3 replicated from the element 6 of the second register;

the element 4 replicated from the element 0 of the first register;
the element 5 replicated from the element 2 of the first register;
the element 6 replicated from the element 4 of the first register; and
the element 7 replicated from the element 6 of the first register.

37. The method as recited in Claim 14, wherein the subset is comprised of the elements 4, 5, 6, and 7 from the first register and the elements 4, 5, 6, and 7 from the second register.

38. The method as recited in Claim 37, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 4 of the second register;
the element 1 replicated from the element 4 of the first register;
the element 2 replicated from the element 5 of the second register;
the element 3 replicated from the element 5 of the first register;
the element 4 replicated from the element 6 of the second register;
the element 5 replicated from the element 6 of the first register;
the element 6 replicated from the element 7 of the second register; and
the element 7 replicated from the element 7 of the first register.

39. The method as recited in Claim 14, wherein the subset is comprised of the elements 0, 1, 2, and 3 from the first register and the elements 0, 1, 2, and 3 from the second register.

40. The method as recited in Claim 39, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 0 of the second register;

the element 1 replicated from the element 0 of the first register;

the element 2 replicated from the element 1 of the second register;

the element 3 replicated from the element 1 of the first register;

the element 4 replicated from the element 2 of the second register;

the element 5 replicated from the element 2 of the first register;

the element 6 replicated from the element 3 of the second register; and

the element 7 replicated from the element 3 of the first register.

41. The method as recited in Claim 14, wherein the subset is comprised of the elements 4, 5, 6, and 7 from the first register.

42. The method as recited in Claim 41, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 4 of the first register;

the element 2 replicated from the element 5 of the first register;

the element 4 replicated from the element 6 of the first register;

the element 6 replicated from the element 7 of the first register; and

the elements 1, 3, 5, and 7 containing a zero in all the bits.

43. The method as recited in Claim 14, wherein the subset is comprised of the elements 0, 1, 2, and 3 from the first register.

44. The method as recited in Claim 43, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 0 of the first register;

the element 2 replicated from the element 1 of the first register;

the element 4 replicated from the element 2 of the first register;

the element 6 replicated from the element 3 of the first register; and the elements 1, 3, 5, and 7 containing a zero in all the bits.

45. The method as recited in Claim 14, wherein the subset is comprised of the elements 4, 5, 6, and 7 from the first register.

46. The method as recited in Claim 45, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 4 of the first register;

the element 1 replicating the sign bit of the element 4 of the first register in all the bits;

the element 2 replicated from the element 5 of the first register;

the element 3 replicating the sign bit of the element 5 of the first register in all the bits;

the element 4 replicated from the element 6 of the first register;

the element 5 containing the sign bit of the element 6 of the first register in all the bits;

the element 6 replicated from the element 7 of the first register; and

the element 7 containing the sign bit of the element 7 of the first register in all the bits.

47. The method as recited in Claim 14, wherein the subset is comprised of the elements 0, 1, 2, and 3 from the first register.

48. The method as recited in Claim 47, wherein the particular order of the elements in the third register comprises:

the element 0 replicated from the element 0 of the first register;

the element 1 containing the sign bits of the element 0 of the first register;

the element 2 replicated from the element 1 of the first register;

the element 3 containing the sign bits of the element 1 of the first register;

the element 4 replicated from the element 2 of the first register;

the element 5 containing the sign bits of the element 2 of the first register;

the element 6 replicated from the element 3 of the first register; and

the element 7 containing the sign bits of the element 3 of the first register.

ABSTRACT

The present invention provides alignment and ordering of vector elements for SIMD processing. In the alignment of vector elements for SIMD processing, one vector is loaded from a memory unit into a first register and another vector is loaded from the memory unit into a second register. The first vector contains a first byte of an aligned vector to be generated. Then, a starting byte specifying the first byte of an aligned vector is determined. Next, a vector is extracted from the first register and the second register beginning from the first bit in the first byte of the first register continuing through the bits in the second register. Finally, the extracted vector is replicated into a third register such that the third register contains a plurality of elements aligned for SIMD processing. In the ordering of vector elements for SIMD processing, a first vector is loaded from a memory unit into a first register and a second vector is loaded from the memory unit into a second register. Then, a subset of elements are selected from the first register and the second register. The elements from the subset are then replicated into the elements in the third register in a particular order suitable for subsequent SIMD vector processing.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Van Hook *et al.*

Appl. No.: To be assigned

Filed: Herewith

For: **Alignment and Ordering of Vector Elements for Single Instruction Multiple Data Processing**

Art Unit: To be assigned

Examiner: To be assigned

Atty. Docket: 1778.0100002 (0055.20US)

Letter to PTO Draftsman: Submission of Formal Drawings

Commissioner for Patents
Washington, D.C. 20231

Sir:

Submitted herewith are 17 sheets of formal drawings with Figures 1-5, 6A-6B, 7, 8A-8H, 9, 10A-10H, corresponding to the informal drawings submitted with the above-captioned application. The application number, group art unit and attorney docket number appear on the back of each sheet. Acknowledgment of the receipt, approval, and entry of these formal drawings into this application is respectfully requested.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX P.L.L.C.

Michael B. Ray
Attorney for Applicants
Registration No. 33,997

Date: 9/15/00

1100 New York Avenue, N.W.
Suite 600
Washington, D.C. 20005-3934
(202) 371-2600

P:\USERS\BPORTER\MBR\1778 0100002 Sub of Formal Drawings

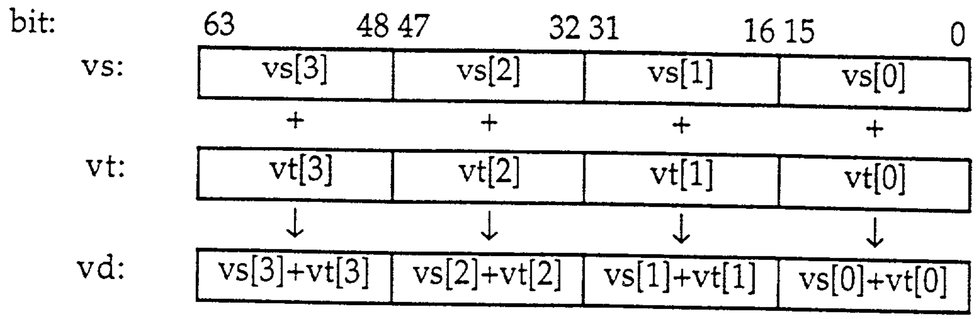

FIG. 1
(Prior Art)

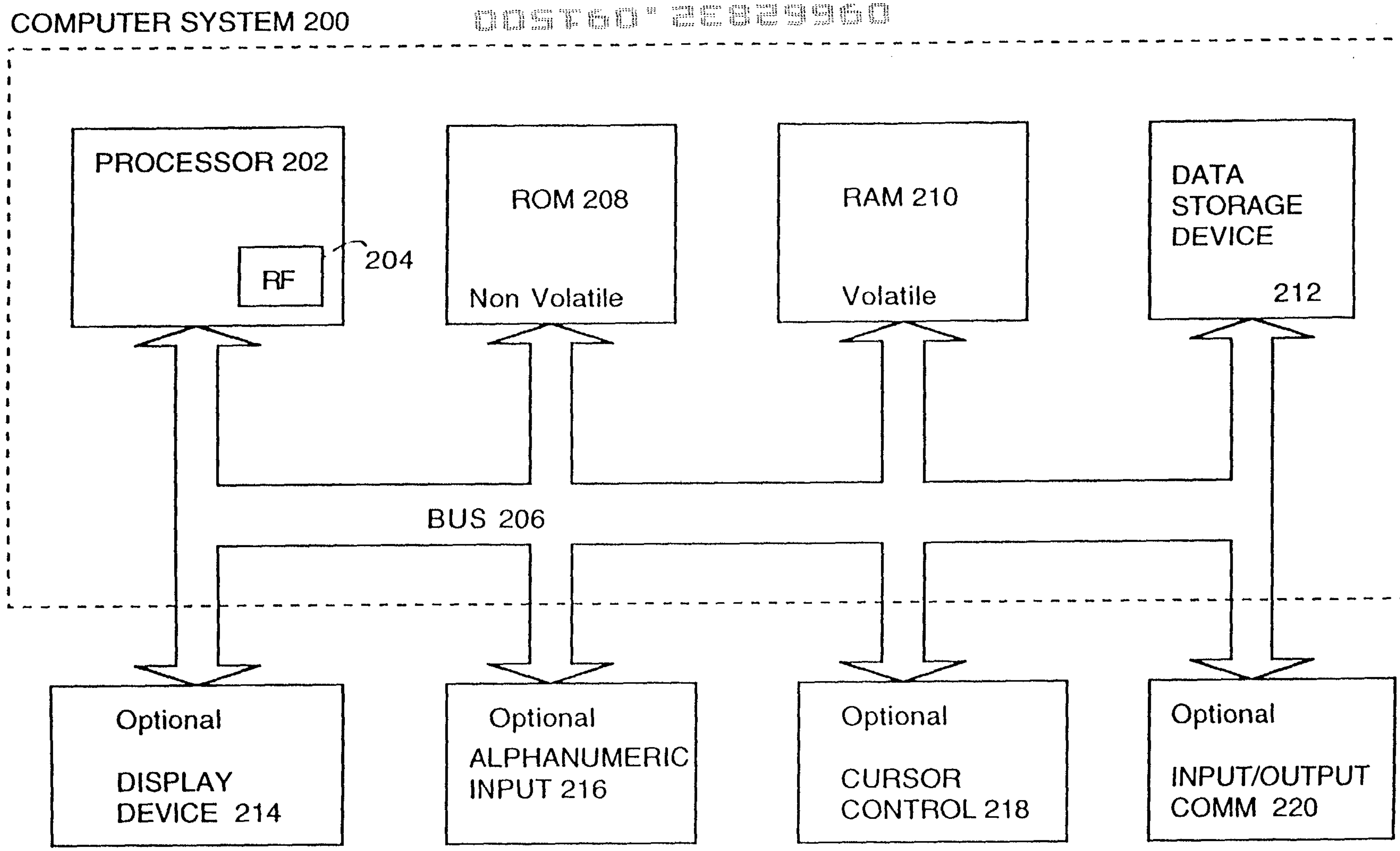
COMPUTER SYSTEM 200
PROCESSOR 202
RF
204
ROM 208
Non Volatile
RAM 210
Volatile
DATA
STORAGE
DEVICE
212
BUS 206
Optional
DISPLAY
DEVICE 214
Optional
ALPHANUMERIC
INPUT 216
Optional
CURSOR
CONTROL 218
Optional
INPUT/OUTPUT
COMM 220

FIG. 2

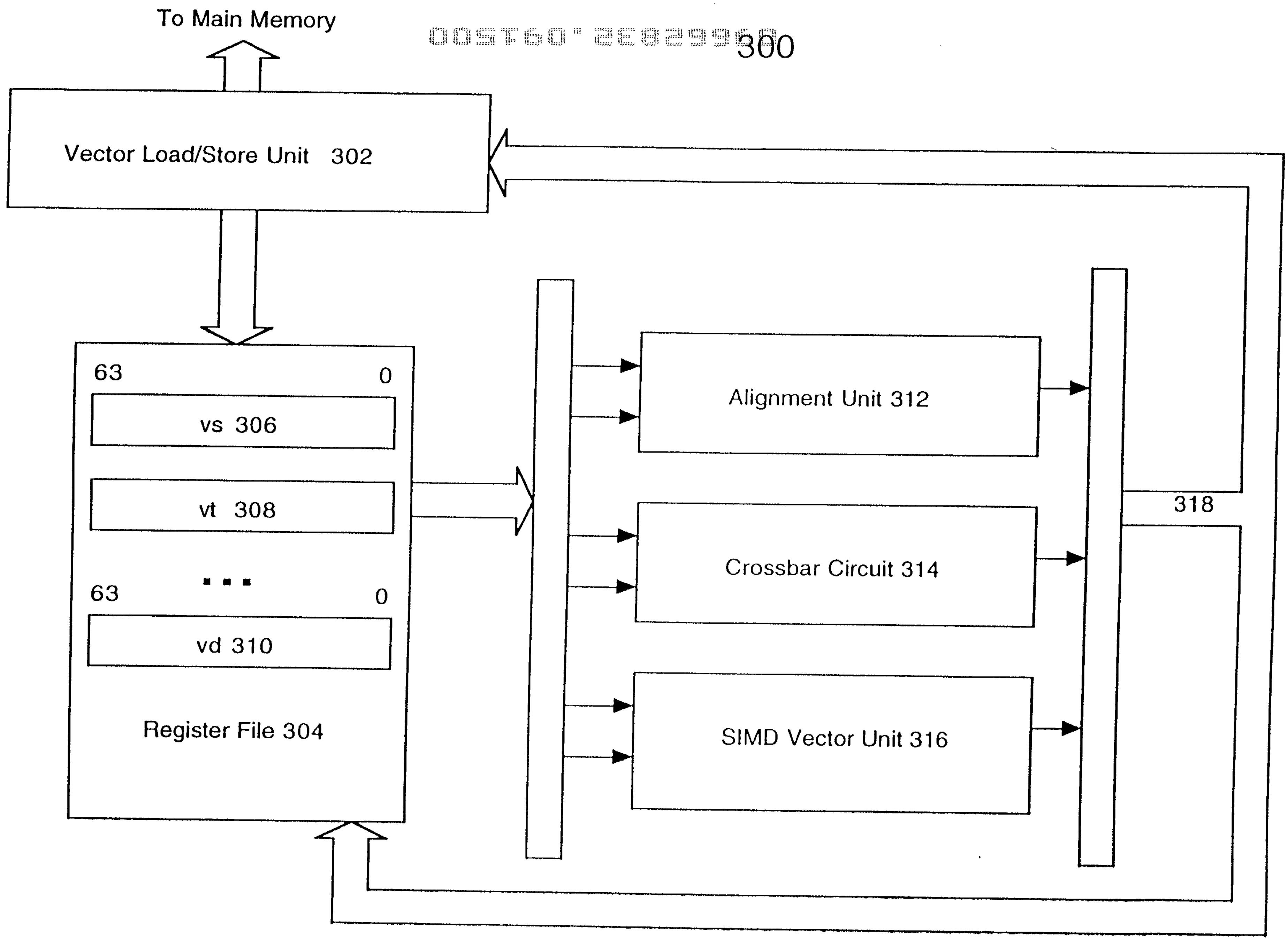
300
To Main Memory
Vector Load/Store Unit 302
63
0
vs 306
vt 308
63
0
vd 310
Register File 304
Alignment Unit 312
Crossbar Circuit 314
SIMD Vector Unit 316
318

FIG. 3

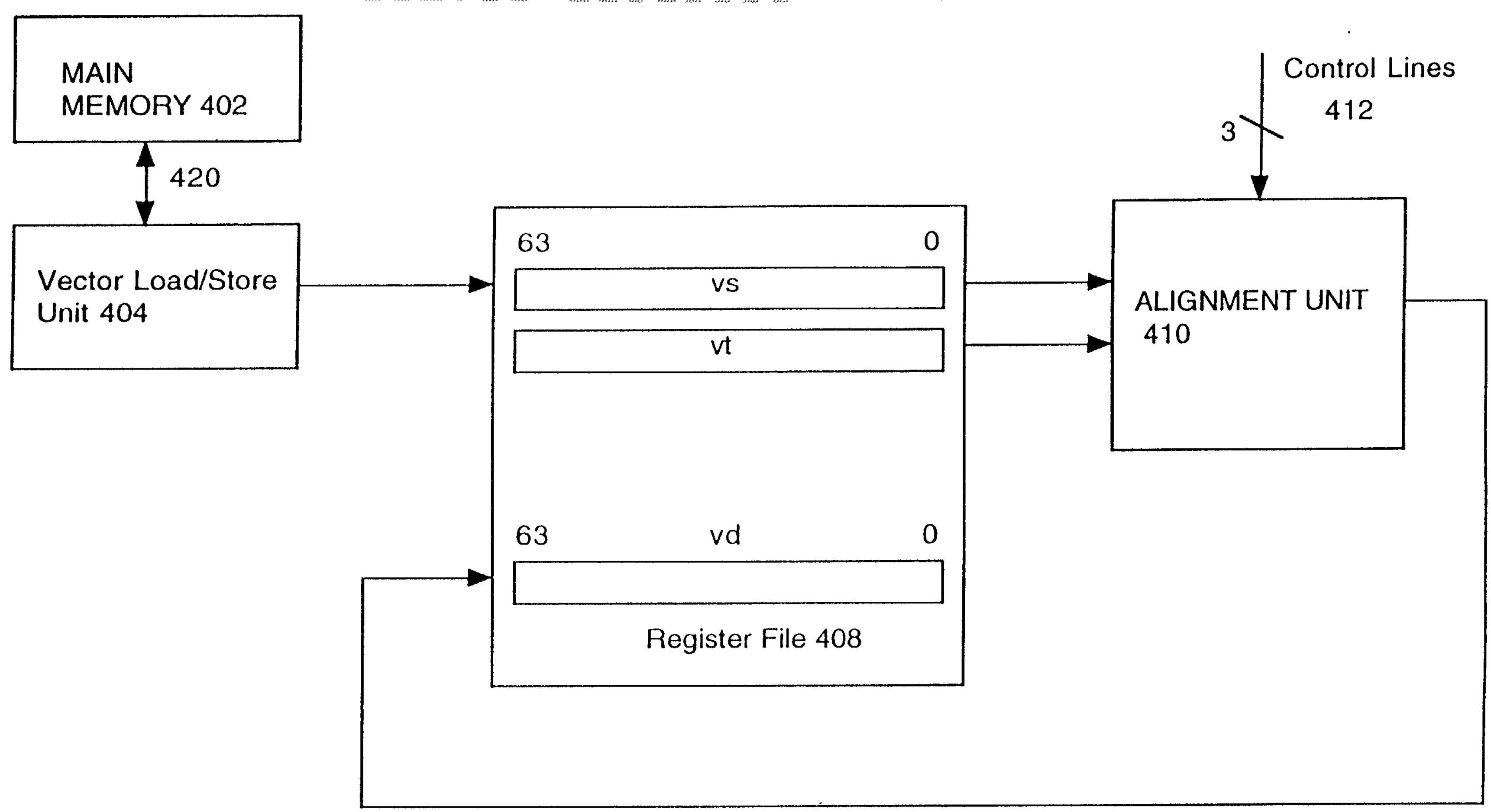
MAIN
MEMORY 402
420
Vector Load/Store
Unit 404
63
0
vs
vt
63
vd
0
Register File 408
Control Lines
412
3
ALIGNMENT UNIT
410

FIG. 4

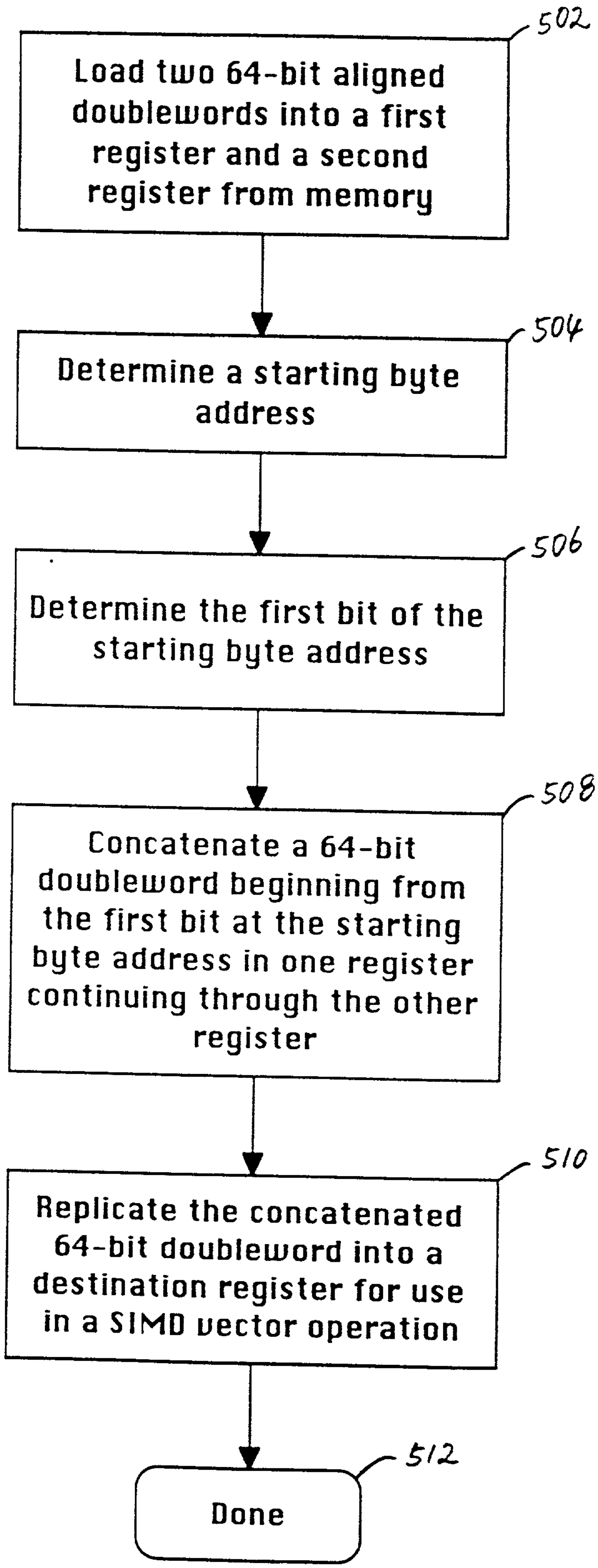
502
Load two 64-bit aligned doublewords into a first register and a second register from memory
504
Determine a starting byte address
506
Determine the first bit of the starting byte address
508
Concatenate a 64-bit doubleword beginning from the first bit at the starting byte address in one register continuing through the other register
510
Replicate the concatenated 64-bit doubleword into a destination register for use in a SIMD vector operation
512
Done

FIG. 5

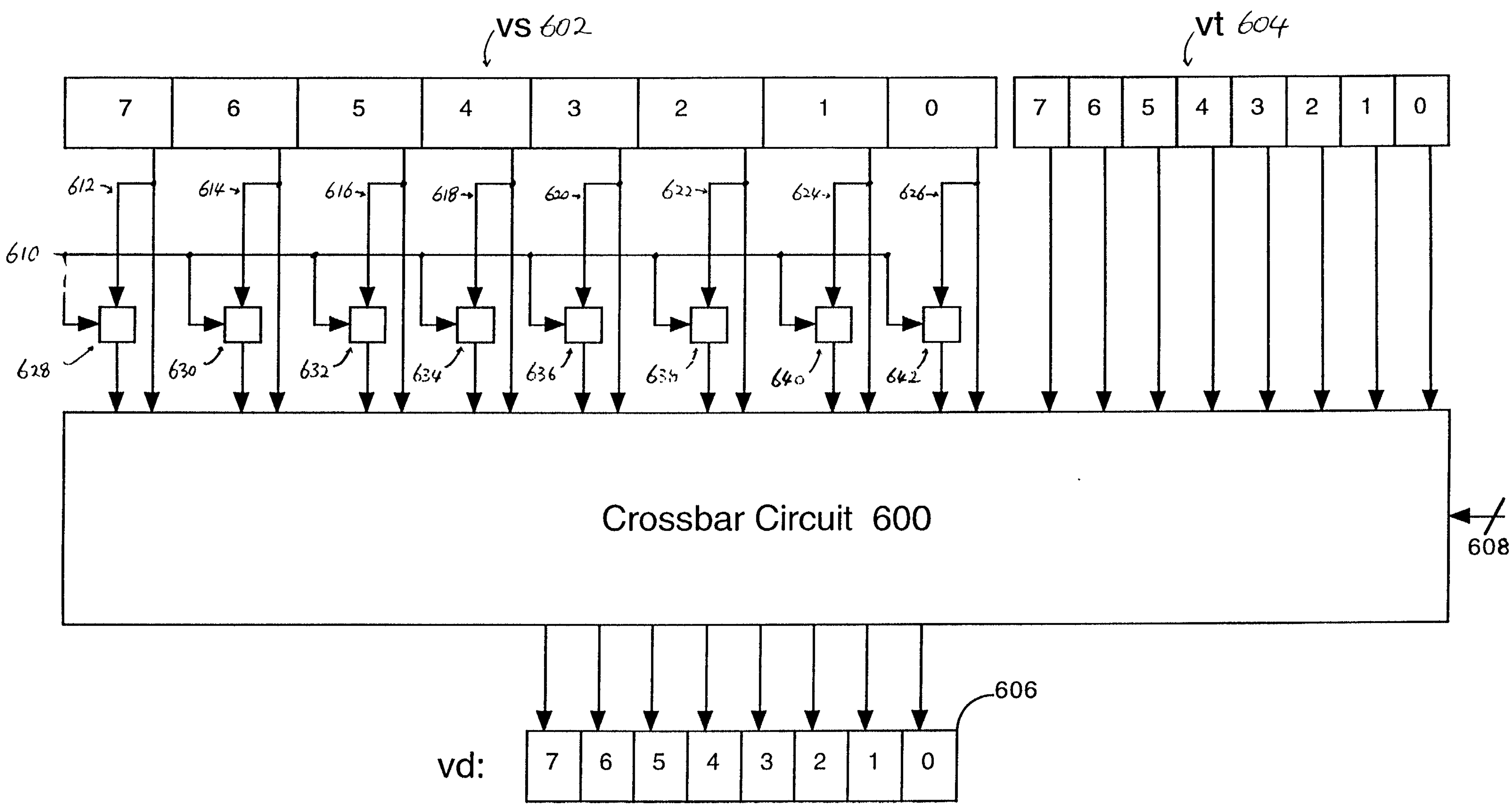
vs 602
vt 604
7
6
5
4
3
2
1
0
612
614
616
618
620
622
624
626
610
628
630
632
634
636
640
642
Crossbar Circuit 600
608
606
vd:

FIG. 6A

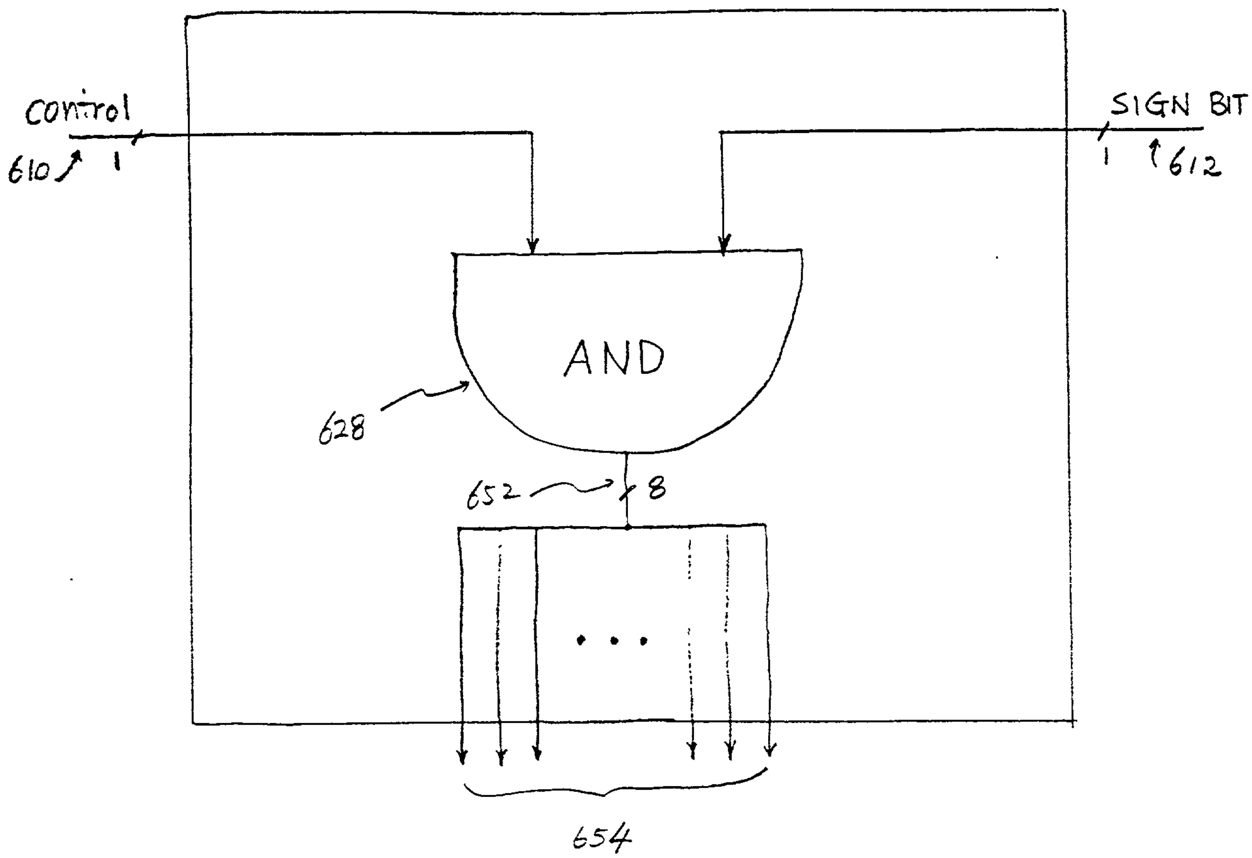

FIG. 6B

63							0	
vd[7]	vd[6]	vd[5]	vd[4]	vd[3]	vd[2]	vd[1]	vd[0]	702
0	vs[7]	0	vs[6]	0	vs[5]	0	vs[4]	704
0	vs[3]	0	vs[2]	0	vs[1]	0	vs[0]	706
sign vs[7]	vs[7]	sign vs[6]	vs[6]	sign vs[5]	vs[5]	sign vs[4]	vs[4]	708
sign vs[3]	vs[3]	sign vs[2]	vs[2]	sign vs[1]	vs[1]	sign vs[0]	vs[0]	710
vs[7]	vs[5]	vs[3]	vs[1]	vt[7]	vt[5]	vt[3]	vt[1]	712
vs[6]	vs[4]	vs[2]	vs[0]	vt[6]	vt[4]	vt[2]	vt[0]	714
vs[7]	vt[7]	vs[6]	vt[6]	vs[5]	vt[5]	vs[4]	vt[4]	716
vs[3]	vt[3]	vs[2]	vt[2]	vs[1]	vt[1]	vs[0]	vt[0]	718

FIG. 7

UPUH

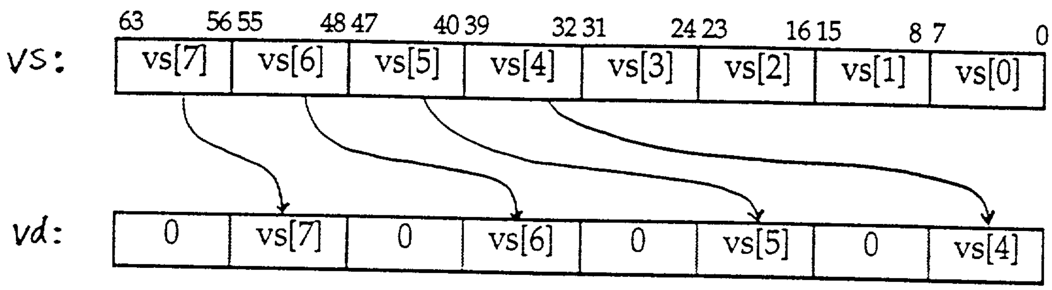

FIG. 8A

UPUL

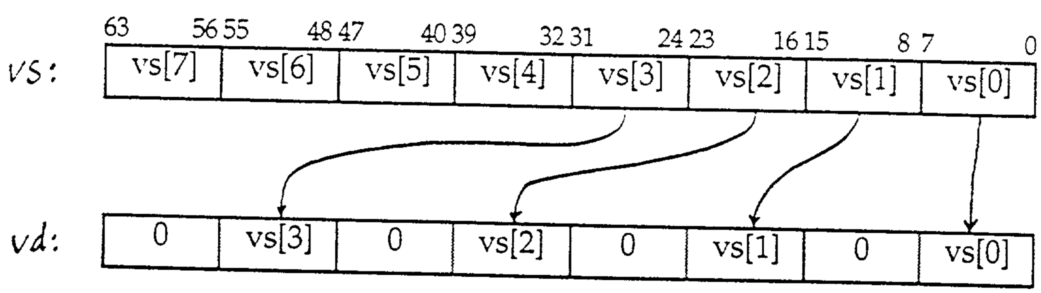

FIG. 8B

UPSH

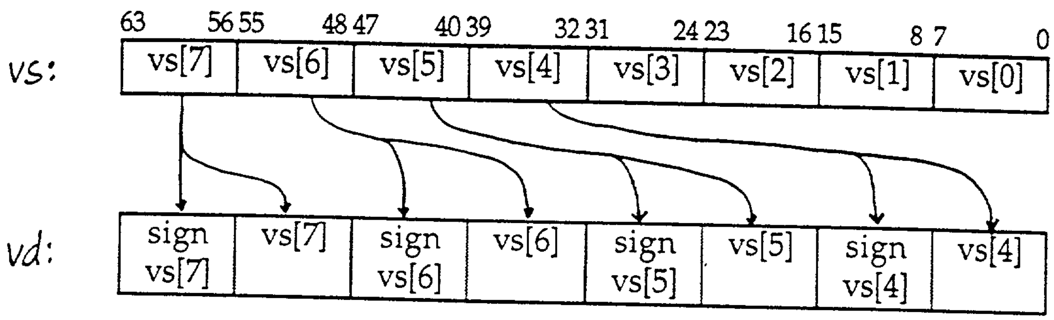

FIG. 8C

UPSL

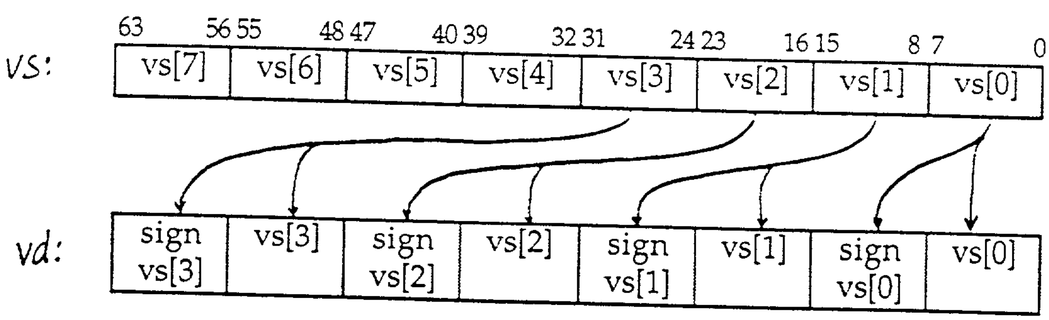

FIG. 8D

PACH

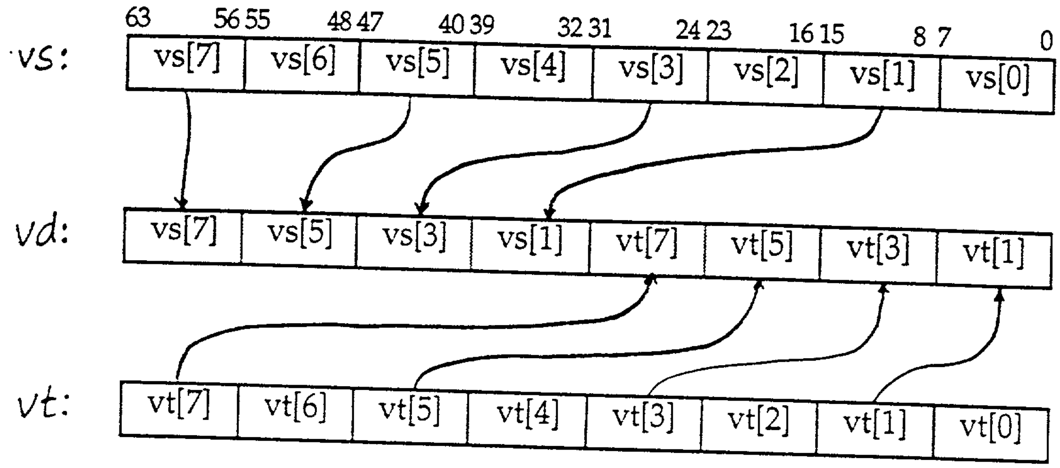

FIG. 8E

PACL

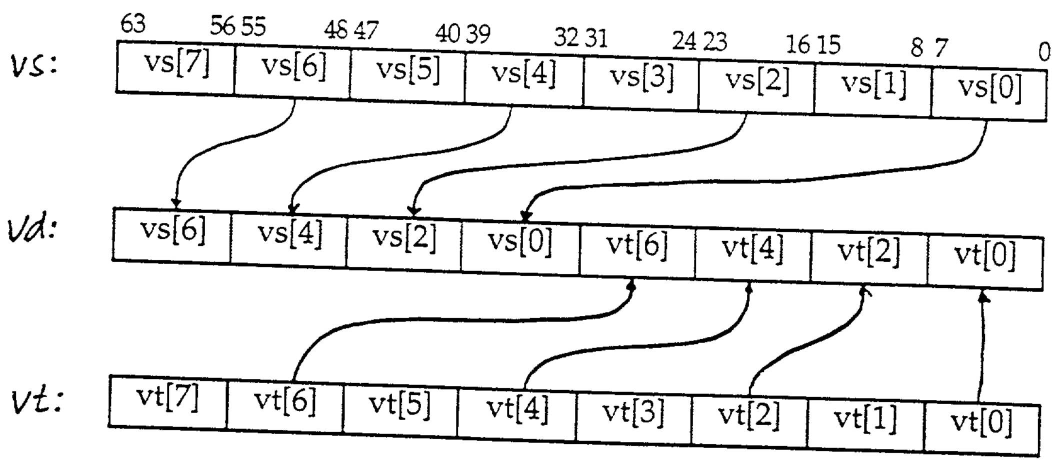

FIG. 8F

MIXH

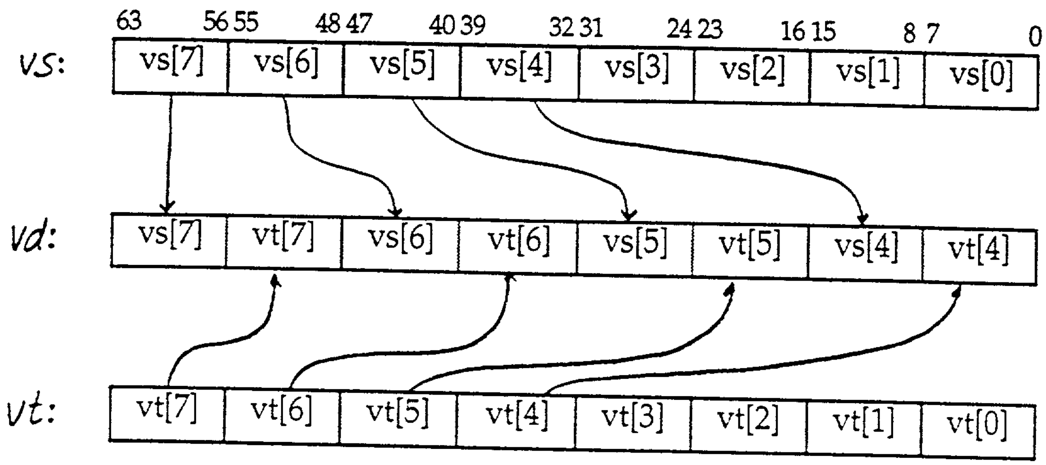

FIG. 8G

MIXL

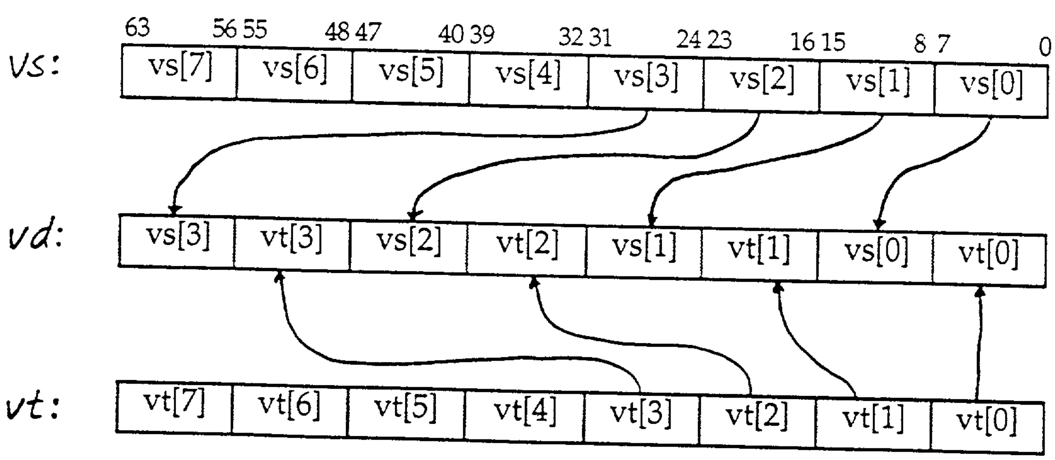

FIG. 8H

63			0	
vd[3]	vd[2]	vd[1]	vd[0]	902
vs[3]	vt[3]	vs[2]	vt[2]	904
vs[1]	vt[1]	vs[0]	vt[0]	906
vs[3]	vs[1]	vt[3]	vt[1]	908
vs[2]	vs[0]	vt[2]	vt[0]	910
vs[2]	vt[3]	vs[0]	vt[1]	912
vs[0]	vt[1]	vs[2]	vt[3]	914
vs[3]	vs[2]	vt[3]	vt[2]	916
vs[1]	vs[0]	vt[1]	vt[0]	918

FIG. 9

MIXH

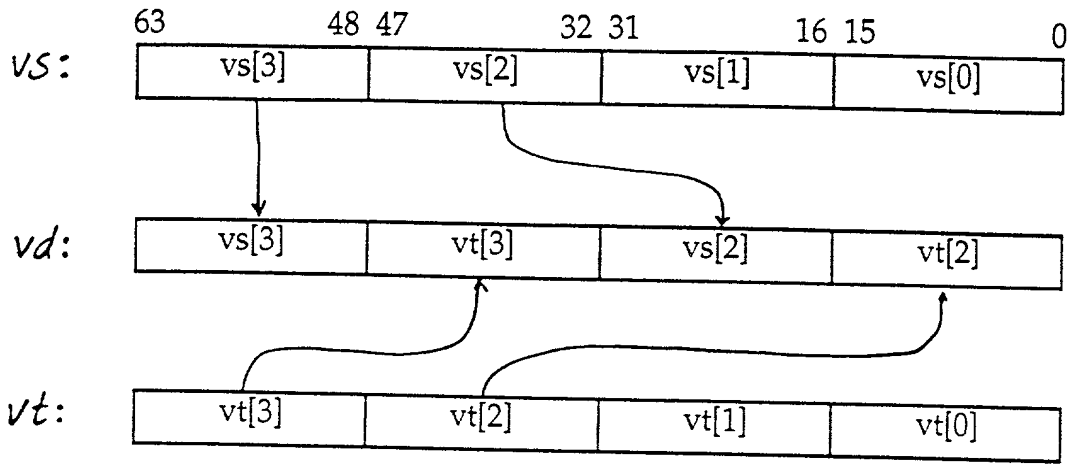

FIG. 10A

MIXL

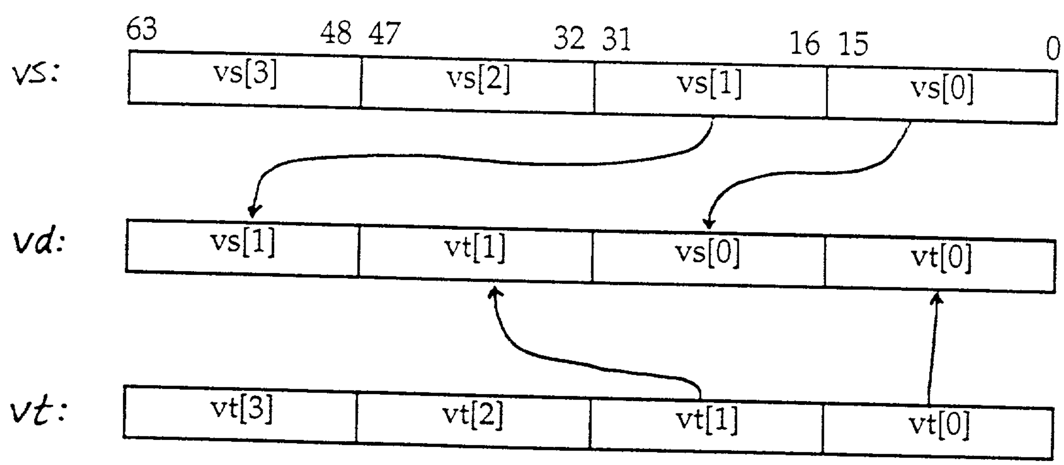

FIG. 10B

PACH

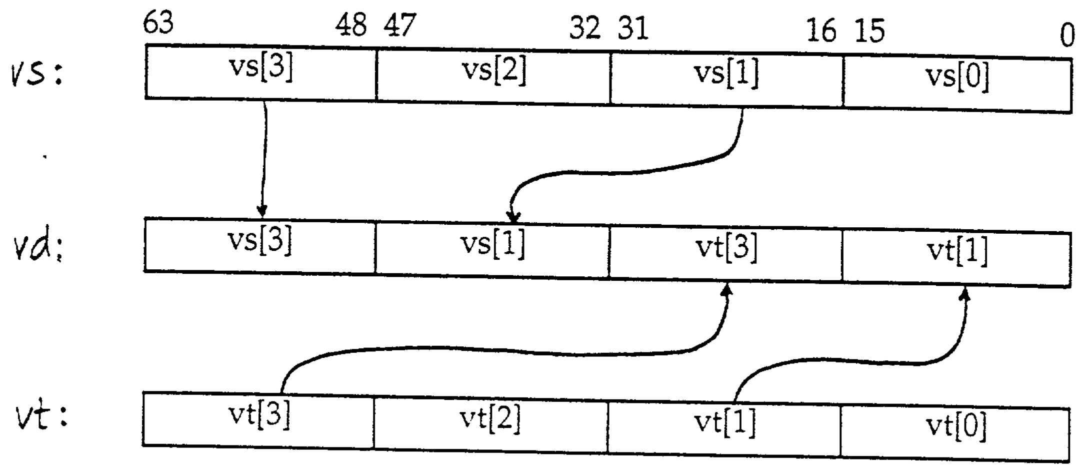

FIG. 10C

PACL

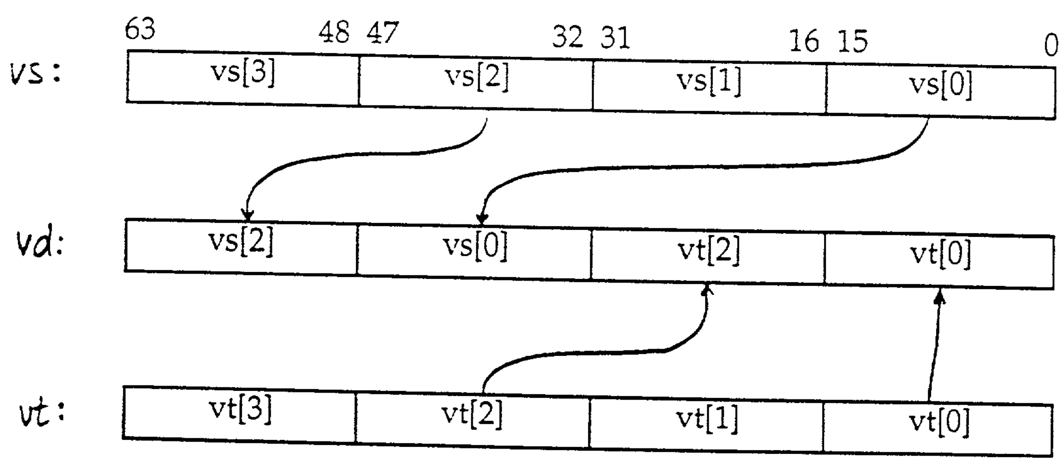

FIG. 10D

BFLA

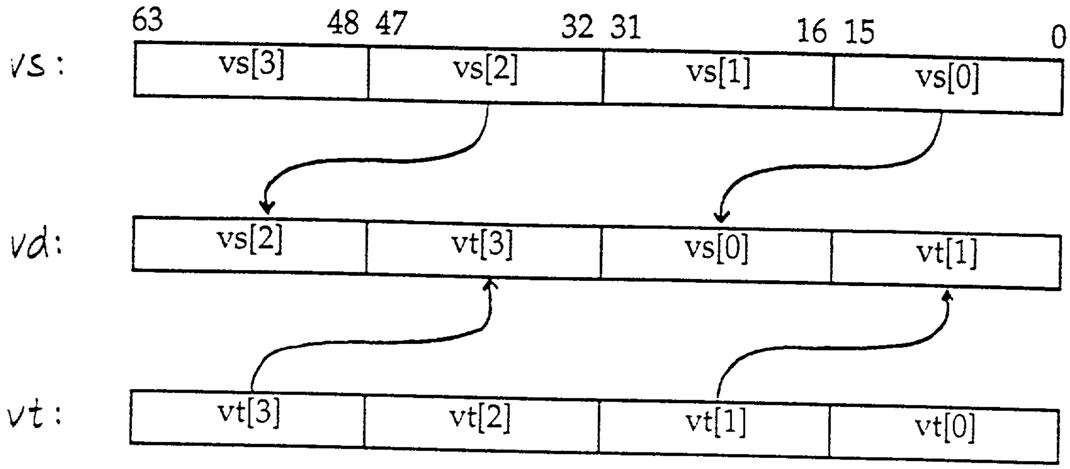

FIG. 10E

BFLB

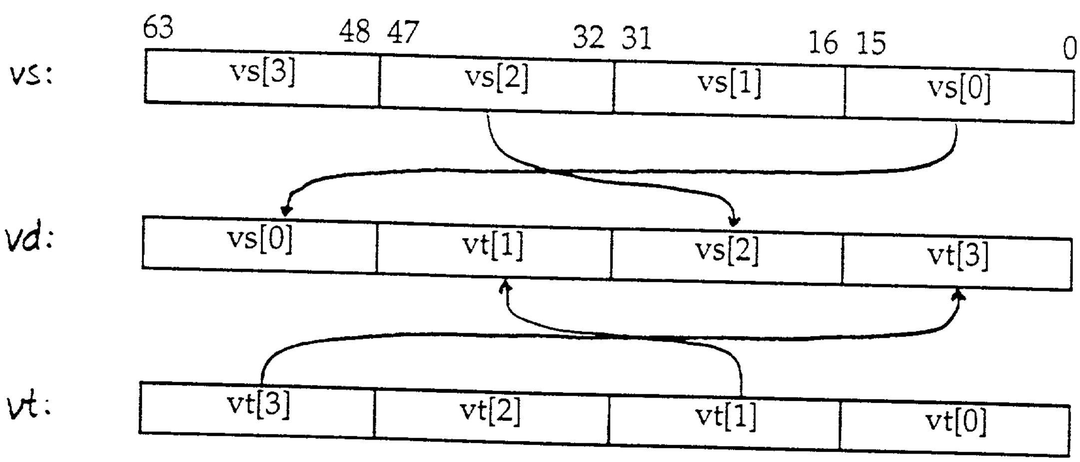

FIG. 10F

REPA

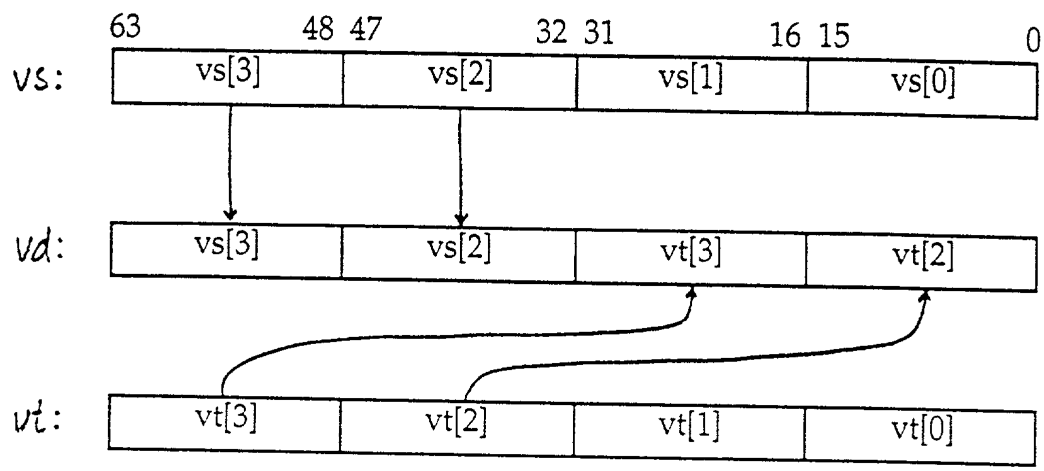

FIG. 10G

REPB

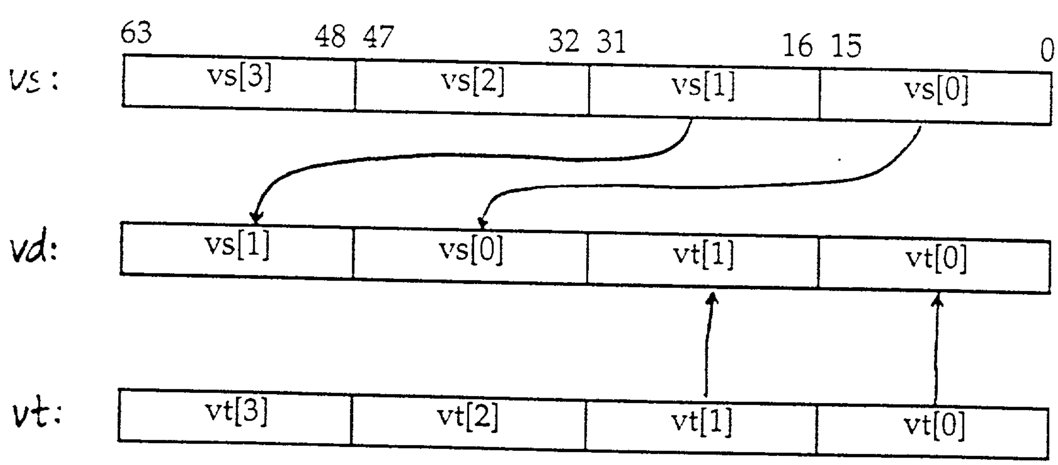

FIG. 10H

Declaration and Power of Attorney for a Patent Application

Declaration

As below named inventor, I hereby declare that my residence post office address, and citizenship are as stated below my name. Further, I hereby declare that I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

the specification of which:

.... is attached hereto, or
X was filed on 10/9/97 as application serial no. 08/947,649 : and
.... was amended on ....

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above; and

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

Foreign Priority Claim

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Date Filed	Priority Claimed	
			.... yes	.... no
			.... yes	.... no

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Serial Number	Filing Date	Status (patented/pending/abandoned)

rev. 1/98 dbp

Power of Attorney

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent Trademark Office connected therewith.

James P. Hao	Registration No.: 36,398
Anthony C. Murabito	Registration No.: 35,295
John P. Wagner	Registration No.: 35,398
Glenn D. Barnes	Registration No.: P-42,293
Wilfred H. Lam	Registration No.: P-41,923
Steve Weiner	Registration No.: 38,330
Chris Byrne	Registration No.: 32,204
Irene Fernandez	Registration No.: 34,625
John Brigden	Registration No.: 40,530

Send Correspondence to:

WAGNER, MURABITO & HAO
Two North Market Street, Third Floor
San Jose, California 95113
(408) 938-9060

Signatures

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: **Timothy J. van Hook**

Inventor's Signature [signature] Date 7/16/98
James P. Hao signing for Timothy J. van Hook under 37 C.F.R. § 1.47 (b)
Residence Atherton, California (City State) Citizenship USA
P.O. Address 224 Oakgrove Avenue, Atherton, California 94027

Full Name of Second/Joint Inventor: **Peter Hsu**

Inventor's Signature Date
Residence Fremont, California (City State) Citizenship
P.O. Address 2853 Welk Common, Fremont, California 94555

Full Name of Third/Joint Inventor: **William A. Huffman**

Inventor's Signature Date
Residence Los Gatos, California (City State) Citizenship USA
P.O. Address 16205 Roseleaf Lane, Los Gatos, California 95032

rev. 1/98 dbp

Full Name of Fourth/Joint Inventor: **Henry P. Moreton**

Inventor's Signature ______________________ Date ____________

Residence Woodside, California Citizenship USA
(City State)

P.O. Address 140 Phillip Road, Woodside, California 94062-2625

Full Name of Fifth/Joint Inventor: **Earl A. Killian**

Inventor's Signature ______________________ Date ____________

Residence Los Altos Hills, California Citizenship USA
(City State)

P.O. Address 27961 Central Drive, Los Altos Hills, California 94022

Declaration and Power of Attorney for a Patent Application

Declaration

As below named inventor, I hereby declare that my residence post office address, and citizenship are as stated below my name. Further, I hereby declare that I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

the specification of which:

______ is attached hereto, or
__X__ was filed on __10/9/97__ as application serial no. __08/947,649__ : and
______ was amended on ______________

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above; and

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

Foreign Priority Claim

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Date Filed	Priority Claimed
			______ yes ______ no
			______ yes ______ no

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Serial Number	Filing Date	Status (patented/pending/abandoned)

rev. 1/98 dbp

Power of Attorney

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent Trademark Office connected therewith.

James P. Hao	Registration No.: 36,398
Anthony C. Murabito	Registration No.: 35,295
John P. Wagner	Registration No.: 35,398
Glenn D. Barnes	Registration No.: P-42,293
Wilfred H. Lam	Registration No.: P-41,923
Steve Weiner	Registration No.: 38,330
Chris Byrne	Registration No.: 32,204
Irene Fernandez	Registration No.: 34,625
John Brigden	Registration No.: 40,530

Send Correspondence to:

WAGNER, MURABITO & HAO
Two North Market Street, Third Floor
San Jose, California 95113
(408) 938-9060

Signatures

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: Timothy J. van Hook

Inventor's Signature ____________ Date ____________
Residence Atherton, California Citizenship USA
(City State)
P.O. Address 224 Oakgrove Avenue, Atherton, California 94027

Full Name of Second/Joint Inventor: Perter Hsu

Inventor's Signature [signature] Date 3/4/98
Residence Fremont, California Citizenship ____________
(City State)
P.O. Address 2853 Welk Common, Fremont, California 94555

Full Name of Third/Joint Inventor: William A. Huffman

Inventor's Signature ____________ Date ____________
Residence Los Gatos, California Citizenship USA
(City State)
P.O. Address 16205 Roseleaf Lane, Los Gatos, California 95032

Full Name of Fourth/Joint Inventor: **Henry P. Moreton**

Inventor's Signature .... Date ....

Residence Woodside, California Citizenship USA
(City State)

P.O. Address 140 Phillip Road, Woodside, California 94062-2625

Full Name of Fifth/Joint Inventor: **Earl A. Killian**

Inventor's Signature .... Date ....

Residence Los Altos Hills, California Citizenship USA
(City State)

P.O. Address 27961 Central Drive, Los Altos Hills, California 94022

Declaration and Power of Attorney for a Patent Application

Declaration

As below named inventor, I hereby declare that my residence post office address, and citizenship are as stated below my name. Further, I hereby declare that I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

the specification of which:

___ is attached hereto, or
X was filed on 10/9/97 as application serial no. 08/947,649 : and
___ was amended on ___

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above; and

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

Foreign Priority Claim

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Date Filed	Priority Claimed
			___ yes ___ no
			___ yes ___ no

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Serial Number	Filing Date	Status (patented/pending/abandoned)

Power of Attorney

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent Trademark Office connected therewith.

James P. Hao	Registration No.: 36,398
Anthony C. Murabito	Registration No.: 35,295
John P. Wagner	Registration No.: 35,398
Glenn D. Barnes	Registration No.: P-42,293
Wilfred H. Lam	Registration No.: P-41,923
Steve Weiner	Registration No.: 38,330
Chris Byrne	Registration No.: 32,204
Irene Fernandez	Registration No.: 34,625
John Brigden	Registration No.: 40,530

Send Correspondence to:

WAGNER, MURABITO & HAO
Two North Market Street, Third Floor
San Jose, California 95113
(408) 938-9060

09582305

Signatures

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: **Timothy J. van Hook**

Inventor's Signature .... Date ....
Residence Atherton, California (City State) Citizenship USA
P.O. Address 224 Oakgrove Avenue, Atherton, California 94027

Full Name of Second/Joint Inventor: **Perter Hsu**

Inventor's Signature .... Date ....
Residence Fremont, California (City State) Citizenship ....
P.O. Address 2853 Welk Common, Fremont, California 94555

Full Name of Third/Joint Inventor: **William A. Huffman**

Inventor's Signature William A. Huffman Date Nov 5, 1998
Residence Los Gatos, California (City State) Citizenship USA
P.O. Address 16205 Roseleaf Lane, Los Gatos, California 95032

rev. 1/98 dbp

Full Name of Fourth/Joint Inventor: **Henry P. Moreton**

Inventor's Signature ______________________ Date ____________

Residence Woodside, California Citizenship USA

(City State)

P.O. Address 140 Phillip Road, Woodside, California 94062-2625

Full Name of Fifth/Joint Inventor: **Earl A. Killian**

Inventor's Signature ______________________ Date ____________

Residence Los Altos Hills, California Citizenship USA

(City State)

P.O. Address 27961 Central Drive, Los Altos Hills, California 94022

Declaration and Power of Attorney for a Patent Application

Declaration

As below named inventor, I hereby declare that my residence post office address, and citizenship are as stated below my name. Further, I hereby declare that I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

the specification of which:

.... is attached hereto, or
X was filed on 10/9/97 as application serial no. 08/947,649 : and
.... was amended on ....

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above; and

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

Foreign Priority Claim

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Date Filed	Priority Claimed
			.... yes .... no
			.... yes .... no

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Serial Number	Filing Date	Status (patented/pending/abandoned)

rev. 1/98 dbp

Power of Attorney

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent Trademark Office connected therewith.

James P. Hao	Registration No.: 36,398
Anthony C. Murabito	Registration No.: 35,295
John P. Wagner	Registration No.: 35,398
Glenn D. Barnes	Registration No.: P-42,293
Wilfred H. Lam	Registration No.: P-41,923
Steve Weiner	Registration No.: 38,330
Chris Byrne	Registration No.: 32,204
Irene Fernandez	Registration No.: 34,625
John Brigden	Registration No.: 40,530

Send Correspondence to:

WAGNER, MURABITO & HAO
Two North Market Street, Third Floor
San Jose, California 95113
(408) 938-9060

Signatures

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: **Timothy J. van Hook**

Inventor's Signature ______________ Date ______
Residence Atherton, California Citizenship USA
(City State)
P.O. Address 224 Oakgrove Avenue, Atherton, California 94027

Full Name of Second/Joint Inventor: **Perter Hsu**

Inventor's Signature ______________ Date ______
Residence Fremont, California Citizenship ______
(City State)
P.O. Address 2853 Welk Common, Fremont, California 94555

Full Name of Third/Joint Inventor: **William A. Huffman**

Inventor's Signature ______________ Date ______
Residence Los Gatos, California Citizenship USA
(City State)
P.O. Address 16205 Roseleaf Lane, Los Gatos, California 95032

rev. 1/98 dbp

Full Name of Fourth/Joint Inventor: Henry P. Moreton

Inventor's Signature [signature] Date 3/6/98

Residence Woodside, California Citizenship USA
(City State)

P.O. Address 140 Phillip Road, Woodside, California 94062-2625

Full Name of Fifth/Joint Inventor: Earl A. Killian

Inventor's Signature Date

Residence Los Altos Hills, California Citizenship USA
(City State)

P.O. Address 27961 Central Drive, Los Altos Hills, California 94022

Declaration and Power of Attorney for a Patent Application

Declaration

As below named inventor, I hereby declare that my residence post office address, and citizenship are as stated below my name. Further, I hereby declare that I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

ALIGNMENT AND ORDERING OF VECTOR ELEMENTS FOR SINGLE INSTRUCTION MULTIPLE DATA PROCESSING

the specification of which:

____ is attached hereto, or
__X__ was filed on __10/9/97__ as application serial no. __08/947,649__ : and
____ was amended on ________

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above; and

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56(a).

Foreign Priority Claim

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Number	Country	Date Filed	Priority Claimed	
________	________	________	____ yes	____ no
________	________	________	____ yes	____ no

U.S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Serial Number	Filing Date	Status (patented/pending/abandoned)
________	________	________
________	________	________

rev. 1/98 dbp

Power of Attorney

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent Trademark Office connected therewith.

James P. Hao	Registration No.: 36,398
Anthony C. Murabito	Registration No.: 35,295
John P. Wagner	Registration No.: 35,398
Glenn D. Barnes	Registration No.: P-42,293
Wilfred H. Lam	Registration No.: P-41,923
Steve Weiner	Registration No.: 38,330
Chris Byrne	Registration No.: 32,204
Irene Fernandez	Registration No.: 34,625
John Brigden	Registration No.: 40,530

Send Correspondence to:

WAGNER, MURABITO & HAO
Two North Market Street, Third Floor
San Jose, California 95113
(408) 938-9060

Signatures

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: **Timothy J. van Hook**

Inventor's Signature ______ Date ______
Residence Atherton, California Citizenship USA
(City State)
P.O. Address 224 Oakgrove Avenue, Atherton, California 94027

Full Name of Second/Joint Inventor: **Peter Hsu**

Inventor's Signature ______ Date ______
Residence Fremont, California Citizenship ______
(City State)
P.O. Address 2853 Welk Common, Fremont, California 94555

Full Name of Third/Joint Inventor: **William A. Huffman**

Inventor's Signature ______ Date ______
Residence Los Gatos, California Citizenship USA
(City State)
P.O. Address 16205 Roseleaf Lane, Los Gatos, California 95032

rev. 1/98 dbp

Full Name of Fourth/Joint Inventor: **Henry P. Moreton**

Inventor's Signature .... Date ....

Residence Woodside, California Citizenship USA

(City State)

P.O. Address 140 Phillip Road, Woodside, California 94062-2625

Full Name of Fifth/Joint Inventor: **Earl A. Killian**

Inventor's Signature [signature] Date 13 July 1998

Residence Los Altos Hills, California Citizenship USA

(City State)

P.O. Address 27961 Central Drive, Los Altos Hills, California 94022